НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

До захисту допущено

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)

"___" _____ 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Спеціалізовані комп'ютерні системи»

спеціальності **123 «Комп'ютерна інженерія»**

на тему: Клієнт-серверний додаток мережі кінотеатрів на основі REST мікросервісів

Виконав:

студент IV курсу, групи КВ-62
2(шифр групи)

Поліщук Андрій Валерійович _____ _____
(прізвище, ім'я, по батькові) (підпис)

Керівник, к.т.н. доц.каф.СПСКС, Потапова К.Р. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент, к.т.н. доц.каф.ПМ, Вовк Л.Б. _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Спеціалізовані комп'ютерні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
«___» _____2020 р.

ЗАВДАННЯ
на дипломний проєкт студента

Поліщук Андрій Валерійович _____
(прізвище, ім'я, по батькові)

1. Тема проєкту Клієнт-серверний додаток мережі кінотеатрів на основі REST мікросервісів _____
_____
_____,
керівник проєкту Потапова Катерина Романівна, к.т.н. доц.каф.СПСКС ____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «25» травня 2020 р. № N1181-С

2. Термін подання студентом проєкту 8 червня 2020 р._____

3. Вихідні дані до проєкту див. Технічне завдання. _____
_____

4. Зміст пояснювальної записки аналіз існуючих систем веб-додатків; платформи та технології для розробки веб додатків; обґрунтування вибору технології, опис розробки додатка, новизна використаних рішень та підходів, висновки _____
_____
_____

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо): Структура таблиць бази даних. Структурна схема; Генерація JWT токену. Схема алгоритму; Схема компонентів архітектури програми. Структурна схема; Схема роботи системи. Схема алгоритму; Презентація. _____
_____
_____
_____

6. Консультанти розділів проєкту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М., доцент		

7. Дата видачі завдання 22.02.2020._____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
	Розроблення та узгодження технічного завдання	22.02.2020	
	Аналіз існуючих рішень	05.04.2020	
	Визначення основних цілей, які необхідно досягти в проєкті	12.04.2020	
	Декомпозиція предметної області	15.04.2020	
	Проектування архітектури та взаємозв'язків системи	18.04.2020	
	Кодування	22.04.2020	
	Тестування	14.05.2020	
	Підготовка матеріалів розділів дипломного проєкту	15.05.2020	
	Підготовка графічної частини дипломного проекту	15.05.2020	
	Підготовка звіту дипломного проекту	15.05.2020	
	Попередній огляд матеріалів диплому на кафедрі	25.05.2020	

Студент _____ Андрій ПОЛІЩУК
(підпис)

Керівник проєкту _____ Катерина ПОТАПОВА
(підпис)

* Консультантом не може бути зазначено керівника дипломного проєкту.

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (56с., 40 рис., 4 додатки).

Об’єкт розробки – Клієнт-серверний додаток мережі кінотеатрів на основі REST мікросервісів.

Розроблена підсистема дозволяє:

- користувачам додатку переглядати афішу, список сеансів, заповненість залу, бронювати квитки на сеанс. Незареєстрований користувач маже зареєструватися;
- адміністраторам редагувати афішу та список сеансів, переглядати заповненість залу з прив’язкою місць до користувачів;
- створювати окремі клієнти за допомогою API.

Система реалізована з використанням REST мікросервісів. В процесі розробки була використана мова програмування Java з використанням фреймворку Spring Framework та інтегроване середовище Intellij IDEA. Для організації даних була використана СУБД MySQL

В ході виконання дипломного проекту:

- проведено аналіз існуючих рішень в мережі Інтернет;
- проведено аналіз відкритих технологій для задоволення визначених потреб;
- розроблено архітектуру системи.

Використання цієї підсистеми дозволить задовольнити потреби клієнтів та співробітників організації, а використані архітектурні рішення дозволють легко підтримувати та розширювати проект, постійно задовольняючи потреби бізнесу.

Приведені необхідні схеми та документація, підведені підсумки щодо проведеної роботи.

Ключові слова: кінотеатр, мікросервіси, Java, Spring Framework, REST, API, JVM, SOLID.

ABSTRACT

Qualification work includes an explanatory note (56p., 40 fig., 4 appendices).

The object of development is a client-server application of a cinema network based on REST microservices.

The developed subsystem allows:

- users of the application to view the poster, the list of sessions, the occupancy of the hall, to book tickets for the session. An unregistered user can register;

- administrators to edit the poster and the list of sessions, view the occupancy of the hall with the binding of seats to users;

- create individual clients using the API.

The system is implemented using REST microservices. The development process used the Java programming language using the Spring Framework and the Intellij IDEA integrated environment. MySQL database was used for data organization

During the implementation of the diploma project:

- analysis of existing solutions on the Internet;

- the analysis of open technologies for satisfaction of the certain needs is carried out;

- system architecture is developed.

The use of this subsystem will meet the needs of customers and employees of the organization, and the architectural solutions used will allow you to easily maintain and expand the project, constantly meeting the needs of the business.

Necessary schemes and documentation are resulted, summarized concerning the carried-out work.

Keywords: cinema, microservices, Java, Spring Framework, REST, API, JVM, SOLID.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ. 045440.002 ТЗ	Клієнт-серверний додаток	4		
			Мережі кінотеатрів на			
			основі REST мікросервісів.			
			Технічне завдання			
	А4	ІАЛЦ. 045440.003 ТП	Клієнт-серверний додаток	2		
			Мережі кінотеатрів на			
			основі REST мікросервісів.			
			Відомість технічного			
			проекту			
	А4	ІАЛЦ. 045440.004 ПЗ	Клієнт-серверний додаток	55		
			Мережі кінотеатрів на			
			основі REST мікросервісів.			
			Пояснювальна записка			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.001 ОА		
Розробив		Поліщук А.В.			Клієнт-серверний додаток мережі кінотеатрів на основі REST мікросервісів. Опис альбому	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Потапова К.Р.					
Н. контроль		Клятченко Я.М.				НТУУ "КПІ" ФПМ КВ-62	
Затвердив		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А1	ІАЛЦ. 045440.005 Д1	Клієнт-серверний додаток	1		
			Мережі кінотеатрів на			
			основі REST мікросервісів.			
			Структура таблиць бази			
			даних. Структурна схема.			
	А1	ІАЛЦ. 045440.006 Д2	Клієнт-серверний додаток	1		
			Мережі кінотеатрів на			
			основі REST мікросервісів.			
			Генерація JWT токену.			
			Схема алгоритму.			
	А1	ІАЛЦ. 045440.007 Д3	Клієнт-серверний додаток	1		
			Мережі кінотеатрів на			
			основі REST мікросервісів.			
			Схема компонентів			
			Архітектури програми.			
			Структурна схема.			
	А1	ІАЛЦ. 045440.008 Д4	Клієнт-серверний додаток	1		
			Мережі кінотеатрів на			
			основі REST мікросервісів.			
			Схема роботи системи.			
			Схема алгоритму.			

Зміст

1. НАЙМЕНУВАННЯ І ОБЛАСТЬ ЗАСТОСУВАННЯ _____2

2. ПІДСТАВА ДЛЯ РОЗРОБКИ _____2

3. ЦІЛЬ І ПРИЗНАЧЕННЯ РОБОТИ_____2

4. ДЖЕРЕЛА РОБОТИ _____2

5. ТЕХНІЧНІ ВИМОГИ _____3

5.1. Вимоги до системи, що розробляється _____3

5.2. Вимоги до апаратного забезпечення _____3

5.3. Вимоги до мінімального програмного забезпечення _____3

6. ЕТАПИ РОЗРОБКИ _____4

					ІАЛЦ. 045440.002 ТЗ		
Зм.	Арк.	№ докум.	Підп.	Дата			
Розроб.		Поліщук А.В.			Клієнт-серверний додаток мережі кінотеатрів на основі REST мікросервісів Технічне завдання	Літ.	Аркуш 1 / Аркушів 4
Перевір.		Потапова К.Р.					
Н. контр.		Клятченко Я.М.				НТУУ "КПІ" ФПМ КВ-62	
Затв.		Романкевич В.О.					

1. НАЙМЕНУВАННЯ І ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування роботи – «Клієнт-серверний додаток мережі кінотеатрів на основі REST мікросервісів».

Область застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування та спеціалізованих комп'ютерних систем Національного технічного університету України «Київський Політехнічний Інститут».

3. ЦІЛЬ І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є розробка Клієнт-серверний додаток мережі кінотеатрів на основі REST мікросервісів, що повинна слугувати основним інструментом обміну даних в організації та допомагати надавати клієнтам послуги з бронювання квиткиів.

4. ДЖЕРЕЛА РОБОТИ

Джерелами інформації для розроблення є технічна література, публікації у періодичних виданнях та Інтернет ресурси з питань розробки.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до системи, що розробляється

Веб-орієнтована система повинна забезпечувати наступні функції:

- підтримка багатомовності;
- можливість авторизації за допомогою паролю та логіну або електронної пошти;
- розмежування прав доступу для користувачів, поділ на ролі;
- можливість бронювати квитки;
- можливість переглядати наповненість залу;
- можливість переглядати афішу та розклад сеансів;
- можливість адміністраторам вносити зміни у розклад та афішу.
- можливість модернізації системи.

5.2. Рекомендовані вимоги до апаратного забезпечення

- Процесор: Intel Core i7.
- Оперативна пам’ять: 8 Гб.
- Простір на диску: 4 Гб.

5.3. Вимоги до мінімального програмного забезпечення

- Операційна система Windows, Linux чи Mac OS X.
- Комплект розробника застосунків Java Development Kit 8.
- Сервер застосунків Apache Tomcat 9.
- База даних MySQL 6.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Розроблення та узгодження технічного завдання	22.02.2020
2.	Аналіз існуючих рішень	05.04.2020
3.	Визначення основних цілей, які необхідно досягти в проєкті	12.04.2020
4.	Декомпозиція предметної області	15.04.2020
5.	Проектування архітектури та взаємозв’язків системи	18.04.2020
6.	Кодування	22.04.2020
7.	Тестування	14.05.2020
8.	Підготовка матеріалів розділів дипломного проєкту	15.05.2020
9.	Підготовка графічної частини дипломного проекту	15.05.2020
10.	Підготовка звіту дипломного проекту	15.05.2020
11.	Попередній огляд матеріалів диплому на кафедрі	25.05.2020

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
1	А4		Завдання на дипломний	2		
			проєкт.			
2	А4	ДП ІАЛЦ. 045440.001 ОА	Опис Альбому.	2		
3	А4	ДП ІАЛЦ. 045440.002 ТЗ	Технічне завдання.	4		
4	А4	ДП ІАЛЦ. 045440.003 ДП	Відомість дипломного	1		
			проєкту.			
5	А4	ДП ІАЛЦ. 045440.004 ПЗ	Пояснювальна записка.	55		
6	А4	ДП ІАЛЦ. 045440.005 Д1	Структура таблиць бази	1		
			Даних. Структурна			
			схема.			
7	А4	ДП ІАЛЦ. 045440.006 Д1	Генерація JWT токену.	1		
			Схема алгоритму.			
8	А4	ДП ІАЛЦ. 045440.007 Д1	Схема компонентів.	1		
			Структурна схема.			
9	А4	ДП ІАЛЦ. 045440.008 Д1	Схема роботи системи.	1		
			Схема алгоритму.			
10	А4	Додаток 2	Лістинг	20		
11	А4	Додаток 3	Презентація	10		

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.003 ДП		
Розробив		Поліщук А.В.			Клієнт-серверний додаток мережі кінотеатрів на основі REST-мікросервісів. Відомість дипломного проєкту.	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Потапова К.Р.					
Н. контроль		Клятченко Я.М.				НТУУ "КПІ" ФПМ КВ-62	
Затвердив		Романкевич В.О.					

Пояснювальна записка
до дипломного проєкту

на тему: Клієнт-серверний додаток мережі кінотеатрів на основі REST мікросервісів _____

_____

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ 3

ВСТУП 5

1. АНАЛІЗ ІСНУЮЧИХ СИСТЕМ ВЕБ-ДОДАТКІВ 6

1.1 Інтернет та різновиди ресурсів 6

1.2 Аналіз існуючих рішень 10

1.3 Обґрунтування теми дипломного проєту. 16

2. ПЛАТФОРМИ ТА ТЕХНОЛОГІЇ ДЛЯ РОЗРОБКИ ВЕБ-ДОДАТКІВ ... 18

2.1 Java 19

2.2 .NET Framework 24

2.3 Порівняння Java та .NET Framework 25

3. ОБГРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ 27

3.1 Клієнт-серверна архітектура. 28

3.2 Мікросервіси та моноліт 31

3.3 REST та RESTful 35

3.4 Spring Framework 36

3.5 Система управління базами даних MySQL. 39

4. ОПИС РОЗРОБКИ ДОДАТКУ 43

4.1 JWT аутентифікація 43

4.2 Організація даних 46

4.3 Розробка інтерфейсу 47

5. НОВИЗНА ВИКОРИСТАНИХ РІШЕНЬ ТА ПІДХОДІВ 53

5.1 Реактивність. Spring WebFlux 53

ВИСНОВКИ 54

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 55

ДОДАТКИ

Додаток 1. Копії графічного матеріалу.

ІАЛЦ.045440.005 Д1. Структура таблиць баз даних. Структурна схема.

ІАЛЦ. 045440.006 Д2. Генерація JWT токену. Схема алгоритму.

ІАЛЦ. 045440.007 Д3. Схема компонентів архітектури програми. Структурна схема.

ІАЛЦ. 045440.008 Д4. Діаграма класів. Схема алгоритму.

Додаток 2. Фрагменти програмного коду.

Додаток 3. Презентація.

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ACID - Atomicity, Consistency, Isolation, Durability – набір принципів, що описують вимоги до транзакційної системи;

API – Application Programming Interface – описання інтерфейсу взіємодії двох систем;

CLR – Common Language Runtime – середовище виконування байт коду Microsoft .NET;

DAO – Data Access Object – архітектурний шар, який відповідає за збереження та видачу даних з бази даних.

DBCP – Data Base Connection Pool – Пул з’єднань з базою даних.

DI – Dependency Injection – процес впровадження зовнішньої залежності компоненту;

DML – Data Manipulation Language – група мов для вставки, видалення, зміни даних в базі даних.

HTML – HyperText Markup Language – мова розмітки гіпертекстових документів;

HTTP – HyperText Transfer Protocol – протокол прикладного рівня передачі гіпертекстової інформації;

IDE – Integrated development environment – інтегроване середовище розробки;

IoC – Inversion of Control – принцип програмування для зменшення зацеплення в програмах;

JIT – Just In Time – технологія динамічної компіляції в Java.

JDBC – Java DataBase Connectivity – специфікація з’єднання Java з базою даних;

JDK – Java Development Kit – набір інструментів для розробки програм на Java;

JRE – Java Runtime Environment – середовище виконання Java;

JSP – Java Server Pages – Java технологія для створення динамічних сторінок.

JVM – Java Virtual Machine – Java віртуальна машина;

J2EE – Java Enterprise Edition – платформа Java для корпоративних проєктів.

MVC – Model-view-controller – шаблон розділення бізнес логіки та відображення;

ORM – Object Relation Mapping – технологія, що пов’язує реляційну модель з об’єктно-орієнтованою.

POJO – Plain Old Java Object – прості Java об'єкти;

REST - Representational **S**tate Transfer – архітектурний стиль взаємодії компонентів.

SQL – Structured Query Language – декларативна мова запитів у базу даних.

URL – Uniform Resource Locator – уніфікований вказівник ресурсу;

АОП – аспектно-орієнтоване програмування;

БД – база даних;

ООП – Об'єктно-орієнтоване програмування.

SPA – Single Page Application – односторінковий додаток.

ВСТУП

Ми живемо в еру глобалізації та цифрових технологій. З кожним роком з’являється все більше нових пристроїв та платформ. Пропонуються нові послуги. Все більше людей надають перевагу послугам в мережі. Сучасна підприємницька діяльність вже не уявляє собі ведення бізнесу без інтернету. Проте найпопулярнішим інструментом взаємодії між бізнесом та клієнтом залишається веб-браузер. Спеціально розроблений веб-застосунок дозволяє бізнесу вийде далеко за межі свого потенціалу, об’єднуючи людей в різних місцях планети. Веб-застосунки стають все більш важливими з розвитком мобільних пристроїв та стрімким збільшенням кількості підключень та швидкості Інтернету. Підприємства будь-якого розміру потребують унікальних веб-платформ для стрімкого росту та підтримки конкурентоспроможності.

Завданням дипломного проєкту є розробка клієнт-серверного додатку мережі кінотеатрів. Особливістю є використання REST мікросервісів при проєктуванні серверної частини, що дозволяє розділити процес розробки серверної та клієнтської частин, а також створювати різні незалежні клієнти для звичайного браузеру або мобільного додатку чи будь якої іншої платформи. Вирішуються задачі модернізації функціоналу, забезпечення безпеки зберігання та обробки даних, розділення доступу до ресурсів згідно ролі користувача та наданих йому прав доступу, багатомовність тощо. Також обрані архітектурні рішення дозволяють подальше розширення функціоналу, що є обов’язковою властивістю великих корпоративних програм бізнесу що розвивається.

1. АНАЛІЗ ІСНУЮЧИХ СИСТЕМ ВЕБ-ДОДАТКІВ

1.1 Інтернет та різновиди ресурсів

Швидкі темпи зростання всесвітньої павутини, простота підключення та доступна ціна обладнання зробила інтернет головним засобом обміну інформації серед людей на планеті. Інтернет допомагає людям донести свою інформацію до великої кількості зацікавлених споживачів, що було досить складно за відсутності подібних речей. Всесвітня мережа стала новим засобом масової інформації де кожен може пропонувати свою унікальну та вузькоспеціалізовану інформацію та об’єднувати зацікавлених людей. Таким чином стрімке зростання обігу інформації сприяє значному розвитку людства загалом. Сьогодні вже існує багато різних видів інтернет ресурсів які забезпечують різні функціональні можливості. Можна виділити кілька основних груп інтернет-ресурсів по змісту та можливостям.

Статичні сайти – такі сайти цілком складаються із сторінок написаних на HTML, можуть мати стилі CSS і навіть деякі вбудовані скипти, графіку та різні інтерактивні елементи які в свою чергу не впливають на зміст сторінки поки не будуть відображені в браузері клієнта. На сервері ця сторінка ніяк не змінюється та однаково відображується для всіх користувачів даного ресурсу.

Динамічні сайти дозволяють будувати сторінку «на льоту». Побудова відбувається на стороні серверу який має необхідні дані. Таким чином одна і та сама сторінка може виглядати по-різному для різних користувачів в різний час тощо. В більш складних рішеннях дані можуть бути дозавантажені в процесі роботи, що дозволяє перетворює сайт зі статичної сторінки в

повноцінну програму. Сьогодні все більше програмних рішень віддають перевагу такому підходу.

Інформаційні сайти – такі сайти, що надають доступ до різного роду інформації: текстової, графічної, мутимедійної. До таких відноситься більшість існуючих ресурсів та відрізняються змістовною частиною. Вони можуть бути новинними, розважальними, інформаційно-тематичними, агрегатори, державні портали, енциклопедії тощо. Інформаційні сайти головним чином відрізняються темою на якій кожен конкретний спеціалізується.

Портал – це складний, багатофункціональний сайт який агрегує різноманітну інформацію та онлайн-сервіси. Портали це великі та складні проєкти, які складно розробляти та підтримувати, проте вони охоплюють максимальну кількість користувачів. Популярні пошукові системи це сайти саме цього типу.

Спеціалізовані сайти – це сайти які розраховані на конкренту цільову аудиторію та мають спеціалізований функціонал. До них відносяться:

- Персональні сайти – свого роду сайт «візитка» на якому автор надає інформацію про себе, свої інтереси, навички, досвід тощо. Вони часто являються статичними та розміщуються на безкоштовних хостингових системах.
- Блог – розвинена ідея персонального сайту, де інформація розміщується в хронологічній послідовності. Для блогів характерна можливість публікації відгуків. Це робить їх середою мережевого спілкування та надає переваги перед електронною поштою, чатами та групами новин.

- Соціальна мережа – сервіси, основна мета яких є пошук, встановлення контактів та спілкування між користувачами мережі. Чим більше вона налічує активних користувачів, тим популярніше вона стає, оскільки більша кількість людей мають змогу знайти одне одного. Раніше відомі сервіси онлайн-знайомств наразі мають багато спільних рис із соціальними мережами, тому відносяться до цієї категорії.
- Форум – інформаційний майданчик для спільного спілкування між користувачами який працює наступним чином: користувач створює тему для обговорення, інші можуть підтримати та доповнити коментарями в яких виражають свою точку зору. Сьогодні соціальні мережі частково перейняли функцію форуму, проте залишається велика кількість даних ресурсів з дуже цінною інформацією та великою популярністю.
- Інтернет-магазин – найпопулярніший вид ресурсів призначений для так званої електронної комерції. Примітивні реалізації можуть являти собою галерею товарів з цінами та контакти для оформлення замовлення, проте сьогодні більшість інтернет-магазинів орієнтовані саме на надання послуг з купівель в Інтернеті, інтегруючись з платіжним сервісом, який обслуговує банківські перекази та інтернет гроші. Це дозволяє клієнту оплачувати замовлення онлайн, а власнику оброблювати замовлення та отримувати оплату на банківський рахунок. Деякі великі магазини навіть надають особисті послуги з доставки кожним разом пропонуючи все більші та зручніші послуги, щоб залучати клієнтів в умовах сильної конкуренції, що робить інтернет-покупки дуже простими та зручними та змушує веб-технології вдосконалюватися.

- Wiki-сайт – специфічний різновид сайтів, який характеризується можливістю колективного керування інформацією вносячи зміни, приймати або відхиляти правки інших користувачів у спільну інформацію так як це реалізовано в Wikipedia.org.

Веб-застосунки – це серверні програми які вирішують конкретні задачі. Часто їх називають веб-сервісами. Вони можуть бути як самодостатніми, так і виконувати допоміжні функції компонентів інформаційного сайту, надаючи йому елементи інтерактивності або розширюючи функціональні можливості. Це численні пакети онлайн застосунків для офісу, задач дизайну, платіжні системи, системи або підсистеми авторизації які вбудовуються у внутрішню структуру сайту.

Можна виділити кілька типів веб-застосунків, в залежності від різних поєднань основних компонентів системи.

- Backend – обробка даних відбувається основним чином на віддаленій машині яка приймає запити та відповідає на них. Така програма може бути написана на багатьох відомих мовах програмування таких як Java, C#, Python, Ruby, PHP та працювати всередині деякого веб-контейнеру, спілкуючись по HTTP протоколу. Дані зберігаються в базі даних, оброблюються програмою, будується HTML відображення та посилається клієнту в браузер.
- Frontend – програма яка завантажується в браузер та працює в ньому використовуючи всі можливості вбудованого JavaScript. Такий підхід використовується коли не треба зберігати дані користувача довше однієї сесії. Це можуть бути наприклад фоторедактори, ігри, конвертери, калькулятори тощо.
- Single page application - (SPA або односторінковий додаток). Це поєднання попередніх двох типів, коли дані користувача можуть

зберігатися віддалено, обробка даних може відбуватися на обох сторонах та без обов'язкових перезавантажень сторінок. Клієнтська сторона сама запитує необхідні дані у сервера «на льоту».

1.2 Аналіз існуючих рішень

Розглянемо існуючі сайти кінотеатрів.

Сайт kino-butterfly.com.ua

Перший кінотеатр який ми розглядаємо – «Баттерфляй». Достатньо відома в Києві мережа кінотеатрів. Головна сторінка зустрічає користувача своїм мінімалістичним дизайном та червоно-жовтою кольоровою палітрою. Для незареєстрованого користувача доступна можливість перегляду наступних сторінок:

- Новини
- Акції
- Книга відгуків
- Вартість квитків
- Правила
- Фотогалерея
- Наші координати

Нижче перелічені назви всіх шести кінотеатрів, що дає змогу відразу вибрати заклад який цікавить більше всього. Зверху є вхід в особистий кабінет та інтеграція з різними соціальними мережами та месенджерами: Facebook, Viber, YouTube, Telegram, Instagram. Також є внутрішня реклама між панеллю навігації та основним контекстом та в самому низу сторінки. На сайті можна переглянути список фільмів які перебувають в прокаті в даний момент, подивитися офіційний трейлер, переглянути найближчі сеанси. На сторінці

фільму можна побачити основну інформацію, таку як: жанр, хронометраж, актори, стислий опис сюжету, офіційний плакат фільму. Можна також поширити запис в соціальних мережах та обрати інший фільм серед запропонованих.

Процес придбання квитків має інтеграцію з платіжними системами. Можна переглядати зал, кількість вільних місць та конфігурацію сидінь відносно екрану. Це дуже зручно, особливо якщо ви плануєте купити кілька квитків для друзів. Процес реєстрації стандартний. Необхідні дані:

- Електронна пошта
- Пароль
- Прізвище, ім’я та по батькові
- Телефон

Після підтвердження та погодження з правилами кінотеатрами умовами бонусної програми необхідно підтвердити (активувати) обліковий запис за допомогою висланого електронного листа. Доволі звична процедура подвійної верифікації. В особистому кабінеті також доступні всі користувацькі можливості такі як: редагування профілю та особистих даних, зміна паролю, придбання та повернення квитків, а також участь у різних акціях та програмах лояльності. Також сайтом передбачається отримання знижок для пільгових категорій людей, що є великим плюсом. На сайті надані контакти гарячої лінії, тож можна отримати відповіді на цікавлячі питання, а також можна задати їх онлайн.

Клієнтська частина сайту достатньо проста, без складної графіки та скриптів, проте має низку цікавих компонентів. Сайт не є статичний, адже контент, який би потребував постійного ручного редагування, приніс би багато незручностей, а також ведення акаунтів користувачів. Сайт

використовує фреймворк JQuerry, який надає сторінкам динамічності та відкритий фреймворк Google Analytics.

Google Analytics – безкоштовний сервіс для створення детальної статистичної інформації відвідувачів веб-ресурсу. Підтримується та розроблюється Google та є продовженням аналітичної системи Urchin on Demand. Все що необхідно це прикріпити JavaScript код фреймворку до сторінки та вся статистична інформація буде збиратися, завантажуватися та оброблюватися на відповідному сервері Google. Безкоштовна версія дозволяє обробку статистики не більше десяти мілiонів переглядів на місяць. Це великий плюс, адже в умовах бізнесу необхідно знати статистику свого сайту та мати показники на які можна орієнтуватися спостерігаючи за тенденціями та змінами.

До переваг даного ресурсу можна віднести відносну простоту інтерфейсу, інтеграцію з відомими соціальними мережами, платіжними системами для оплати квитків та використання Google Analytics як інструменту збору статистичних даних та варіант верстки для мобільного телефону.

До недоліків можна віднести застарілий дизайн в цілому, наявність реклами та відсутність багатомовності інтерфейсу.

Сайт kino-teatr.ua

На відміну від звичайних сайтів кінотеатрів, kino-teatr.ua виконує функцію агрегатора над багатьма закладами де є можливість обрати конкретний кінотеатр та бронювати квитки на показ в ньому. Тут є можливість створення облікового запису. Необхідно вказати логін, пароль, електронну пошту, мову, місто та ввести капчу. Після чого необхідно підтвердити акаунт в листі, надісланий на вказану електронну пошту.

З функціональних можливостей можна виділити наступне:

- До облікового запису закріплюється обраний нік.
- Користувач може отримувати по e-mail відповіді на повідомлення та повідомлення з тим, на які він підписався.
- Можна оцінювати фільми, кінотеатри, акторів а також відгуки про них.
- Слідкувати за цікавими фільмами.
- Брати участь в конкурсах.
- Отримувати розсилку сайту.
- Можна отримати статус редактора та разом з керівництвом брати участь в роботі: розміщувати свої рецензії, мультимедійні файли до фільмів, фото акторів тощо.

Сайт інтегрований з платіжною системою, що дозволяє прикріпити банківську картку та не вводити дані картки кожен раз по типу технології «Master Pass». Використовуються cookies.

Даний веб-ресурс є прикладом досить складного корпоративного програмного продукту з великою кількістю користувачів, наявністю мобільного додатку, інтеграцією з відомими соціальними мережами та наявністю API.

API (англ. application programming interface) – програмний інтерфейс, описання способів за допомогою яких одна комп’ютерна система обмінюється інформацією з іншою. Може входити в описання якогось інтернет-протоколу, фреймворку. Використовується при написанні застосунків. API це також засіб інтеграції різних систем який визначає функціональність яку надає програма. В індустрії програмного забезпечення спільні стандарти API для стандартної функціональності грають важливу роль, так як вони гарантують що всі

програми, які використовують спільне API будуть працювати однаково добре, або принаймні однаковим способом. Набір інтерфейсів для отримання даних абстрагує розробника від тонкощів реалізації цієї функціональності та надає інструменти для взаємодії між системами. Це значно спрощує процес інтеграції систем які можуть бути написані на зовсім різних платформах, використовуючи різні підходи. Таким чином наприклад команда «backend» розробників за допомогою API розробляє набір інтерфейсів по яким сервер буде видавати дані клієнтам які з ним взаємодіють, а команда «frontend» розробників в свою чергу розробляють незалежно клієнт для веб-браузеру та додаток для мобільної платформи. Це допомагає розпаралелити процес розробки на незалежні потоки виконання.

Переваги:

- Наявність API.
- SPA клієнт з асинхронними запитами.
- Багате різноманіття користувацьких можливостей.
- Багатомовність.
- Інтеграція з соціальними мережами.
- Інтеграція з платіжними системами.
- Наявність мобільного додатку.
- Агрегація великої кількості кінотеатрів в межах одного ресурсу.

Недоліки:

- Наявність великої кількості контекстної реклами.

Сайт kinoukraina.zt.ua

Даний сайт є прикладом достатньо простого програмного рішення в якому у якості серверної мови програмування використовується PHP. Це можна побачити у адресному рядку. Всі ресурси закінчуються на .php.

Простий сайт який є статичним без складного функціоналу. Кінотеатр є невеликим закладом на дві глядацькі зали. На сайті немає можливості зареєструватися та вести активну діяльність прив’язуючись до свого акаунту.

Меню складається з наступних пунктів:

- Афіша.
- Фотогалерея.
- Вакансії.
- Співробітництво.
- Контакти.

Можна переглянути список сеансів, афішу, схему залу та зайняті місця. Також можна переглянути основну інформацію про сам фільм, подивитися трейлер та купити квиток/забронювати місце. Можна помітити, що для онлайн купівлі квитків використовується сторонній сервіс «Вкино» за допомогою якого і вирішується проблема пересилання електронних коштів, генерація квитків з подальшим надсиланням клієнту, тощо.

До переваг можна можна віднести простоту розробки, експлуатації та мінімальну вартість обслуговування подібного ресурсу.

До недоліків відносяться:

- Застарілий дизайн.
- Відсутність можливості створювати власний кабінет.
- Відсутність API.
- Відсутність реферальних систем.
- Відсутність мобільного додатку.

1.3 Обґрунтування теми дипломного проєту.

Сьогодні є велика потреба в сучасних програмних рішеннях для ведення бізнесу та автоматизації вирішення проблем обробки інформації складних систем. Сфера послуг не може уявити собі діяльність без інтеграції з Інтернетом. Кожна організація хоче мати особистий веб-ресурс, за допомогою якого клієнти можуть отримувати послуги, кур'єри обробляти замовлення, служба підтримки мала можливість централізовано обробляти запити, слідкувати за статистикою, тенденціями тощо. Також система повинна задовольняти сучасним вимогам безпеки та гнучкості. Система повинна швидко підлаштовуватися під вимоги бізнесу та не бути надмірно витратною.

Сфера послуг є однією з лідерів по швидкості росту та розвитку. З'являється все більше нових закладів, а інші зосереджують навколо себе все більше клієнтів. З розвитком інформаційних технологій та стрімким зростанням бізнесу, впровадження централізованого, веб-орієнтованого застосунку є необхідністю для подальшого розвитку сфери.

Розроблювана система розрахована задовольнити основні потреби організації кінотеатру, автоматизуючи та спрощуючи корпоративні процедури організації.

Система реалізує наступні функції:

- Реєстрація користувачів.
- Наявність ролей та привілеїв.
- Можливість редагування сайту від імені адміністратора.
- Підтримка різних мов інтерфейсу.
- Перегляд афіші.
- Перегляд списку сеансів.
- Перегляд зали.

- Бронювання білетів.
- Перегляд білетів користувача.
- Повернення білетів.
- Редагування афіші.
- Редагування сеансів.

2. ПЛАТФОРМИ ТА ТЕХНОЛОГІЇ ДЛЯ РОЗРОБКИ ВЕБ-ДОДАТКІВ

Вибір технологій для проєкту це один з важливіших та найвідповідальніших етапів підготовки до подальшої розробки. Існує багато різних платформ, мов програмування, кожна з яких зайняла свою нішу де набула найкращого успіху. Технології швидко змінюються та вдосконалюються. Відомі кілька років тому можуть безнадійно застаріти, звільняючи місце новим. В такій незупинній еволюції досить просто заплутатись. Помилка вибору може коштувати дуже багато ресурсів, адже для довгострокового проєкту неприпустимо змінювати платформу взагалі, а проблеми створені технологіями які швидко втрачають актуальність, невдалими рішеннями в них самих, або припинення підтримування та оновлення можуть взагалі погубити проєкт. Отже необхідно вибрати інструмент який з мінімальними ризиками буде служити багато років поспіль.

Обираючи платформу, будемо орієнтуватися на наступні характеристики:

- Веб-додатки як одна з основних спеціалізацій.
- Кросплатформність.
- Підтримка об’єктно-орієнтованої парадигми.
- Відкритий програмний код. Безкоштовність.
- Наявність великої кількості додаткових бібліотек або фреймворків для роботи з мережею, інтерфейсом, даними тощо.
- Велика кількість документації та спільнот які підтримують технологію.
- Уніфікований доступ до баз даних.

2.1 Java

На даний момент Java є одним з найбільш розповсюджених мов програмування. Вперше з’явилася ще в 1996 році як розробка американської компанії Sun Microsystems з гаслом «Написано раз – працює скрізь». Кросплатформність та зворотна сумісність були головними принципами мови, проте потім від другого довелось відмовитись тому, що зворотна сумісність не дозволяє вирізати невдалі рішення в наступних версіях, що створило велику кількість сміття в мові та бібліотеках. Java знайшла застосування в багатьох напрямках програмного забезпечення від мікроконтролерів до складних, високонавантажених Enterprise систем, веб-порталів та сервісів. Операційна система Android з самого початку була написана на Java як і більшість додатків для неї.

Код на мові Java компілюється в байт-код який однаковий незалежно від платформи. Потім віртуальна машина (JVM), специфічна для конкретної платформи, інтерпретує байт-код в машинні команди. Таким чином розробнику не треба піклуватися за те щоб писати різні версії програми для різних операційних систем, або взагалі апаратних засобів. Замість цього для конкретної платформи – Windows, Linux, Mac OS тощо, розробляється своя JVM, у відповідності до специфікації і вже вона виконує окремо написаний та скомпільований байт-код. Подібний підхід забезпечує апаратну переносимість програм на різних платформах.

Такий підхід часто відносять до недоліків, оскільки це інколи може суттєво зменшити швидкість виконання програм. Дійсно, JVM поступається своїм конкурентам C/C++. В середньому віртуальна машина повільніше у півтора-два рази ніж C/C++, хоча в деяких випадках швидша, а в деяких в 7 разів повільніше. Результат непоганий, як плата за переносимість.

Серйозний комплекс заходів по оптимізації допомогли досягти таких результатів:

- Застосування так званої JIT-компіляції при якому байт-код транслюється в машинний код безпосередньо під час роботи програми з можливістю збереження версій класу в машинному коді.
- Використання платформенно-орієнтованого коду в стандартних бібліотеках. Так званий native-код.
- Деякі апаратні засоби для прискореної обробки байт-коду.

Java є об’єктно-орієнтованою мовою мовою з Сі-подібним синтаксисом та має багато схожого з C, C++, C#. Оскільки сьогодні більшість мов маюсь Сі-подібний синтаксис то з переходом на Java не виникає труднощів.

Для того щоб виконувати програми написані на Java необхідно встановити так званий JRE (Java Runtime Environment). В складі JRE знаходиться віртуальна машина на набір класів для запуску програм. Цього цілком достатньо для запуску готових рішень, але для розробки особистих програм необхідно мати ще «набір розробника», або JDK (Java Development Kit). Він включає в себе JRE, компілятор та набір необхідних утиліт.

Java включає в себе декілька сімейств технологій:

- Java SE - Java Standard Edition, поставляється з компілятором, JVM та API. Краще за все підходить для створення настільних систем.
- Java EE - Java Enterprise Edition, набір мпецифікацій для створення програм корпоративного рівня, серверних систем. З 2017 року переданий до Eclipse Foundation після чого змінили назву на Jakarta EE. Видалені відповідні модулі з пакету SE з 11-ї версії.

- Java ME – набір технологій для платформ з обмеженими обчислювальними можливостями, такими як телефони, мікроконтролери, КПК тощо.

Java використовується як головна мова для розробки програмного забезпечення для платформи Android. Байт-код при цьому відрізняється від звичайного. Розроблено спеціалізовану віртуальну машину Dalvik (починаючи з Android 5.0 Lolipop віртуальна машина ART). Для такої компіляції необхідно використовувати Android SDK (Software development kit) розроблений компанією Google. Android Studio визнана офіційним середовищем розробки для цієї платформи.

Основні можливості платформи:

- Гнучкі можливості обробки виняткових ситуацій.
- Автоматичне керування пам’яттю.
- Широкий набір засобів вводу-виводу.
- Велике різноманіття структур даних.
- Засоби для роботи з мережевим обміном даними.
- Засоби для організації обміну даними по HTTP протоколу.
- Вбудовані засоби багатопоточності.
- Уніфікований доступ до баз даних
- Елементи функціонального програмування: лямбди, замикання, Stream API.

Під час кодування, не треба піклуватися про виділення та вивільнення пам’яті в купі (Heap), так як в JVM є збирач сміття. В свою чергу збирач сміття працює в окремих потоках виконання, керуючись самою віртуальною машиною, відповідальний за виділення пам’яті для об’єкту при виконанні оператора new та звільнення пам’яті якщо посилання на об’єкт втрачено.

Таким чином, об’єкт стає придатним для збирання сміття коли він не доступний по будь-якому посиланню для потоків або статичних посилань. Від моменту коли до об’єкту втрачене посилання зі стеку, а циклічні посилання не враховуються, весь об’єкт, який може бути складений з інших помічається та пам’ять яку він займає буде звільнена на наступній ітерації збору сміття. Об’єкти створюються в частині JVM яка називається Heap, який з точки зору збирача сміття розділений на три секції – Young generation, Old generation, Permanent genetarion (рис. 1).

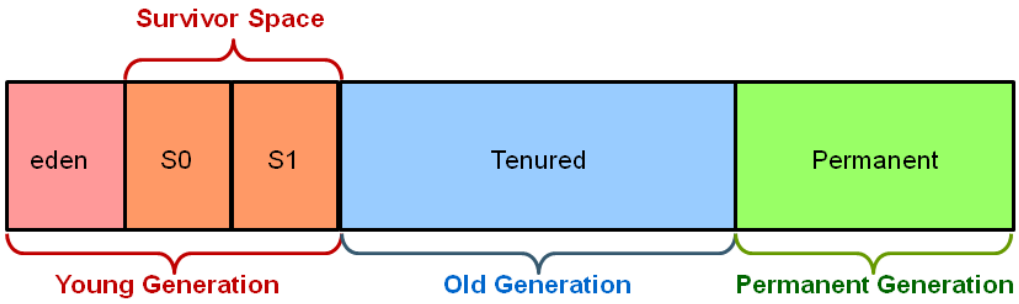

Рис. 1

Об’єкт створюється в розділі Eden. Коли цей розділ переповнюється – активується процедура збору сміття. Пам’ять втрачених об’єктів звільняється, а всі що залишилися – переміщуються в Survivor 1 (на рисунку S0), потім в Survivor 2 після чого об’єкт переноситься в розділ з об’єктами з довгим часом життя – в Old Generation (на рисунку Tenured). Статичні поля класі, самі класи завантажуються в розділ Permanent Generation.

Для роботи за базою даних, в Java використовується так званий JBDC драйвер. Оскільки в світі існує велика кількість різний систем управління

базою даних (СУБД), складно слідкувати за всіма та випускати для них окремий драйвер та підтримувати його. Замість цього було розроблено специфікацію JDBC в якій описано як драйвер мусить функціонувати і вже розробники баз даних, якщо хочуть щоб Java вміла працювати з їх СУБД, розробляють свою версію драйверу. Таким чином для того, щоб мати змогу посилати запити в базу даних, необхідно підключити спеціальний драйвер. Єдина специфікація задає єдиний інтерфейс, таким чином всі драйвери мають абсолютно однаковий набір методів для роботи, що залишає код незмінним при переході на іншу базу даних та інший драйвер, окрім мови запитів, яка не зобов'язана бути єдиною для всіх СУБД тому складності при переході все ж таки є.

Важливою частиною є стандартні структури даних та засоби їх організації. В Java окрім звичайних масивів, широко використовуються колекції (Collection Framework). Це набір інтерфейсів та класів які постійно використовуються та були спроектовані Джошуа Блохом. В основі архітектури лежить інтерфейс Collection в якому визначені основні методи, такі як add(), remove(), toArray(). Collection успадковується від інтерфейсу Iterable.

Інтерфейс List визначає структуру даних «список» та має дві реалізації:

- ArrayList – список на основі масиву.
- LinkedList – структура даних на основі двозв'язного списку.

Інтерфейс Set реалізує концепцію множини і має 2 реалізації:

- HashSet – множина на основі хеш-таблиці.
- TreeSet – множина на основі бінарного дерева.

2.2 .NET Framework

Платформа .NET Framework включає в себе бібліотеку класів та загальномовне середовище виконання CLR. Ядро платформи це CLR. Вона керує кодом під час виконання та надає необхідні служби такі, як керування пам’яттю, керування потоками тощо. Середа накладає умови строгої типізації та інші перевірки які забезпечують надійність та безпеку. Бібліотека класів це комплексний об’єктно-орієнтований набір повторно використовуваних які використовуються для розробки великого різноманіття програм, це і програми які використовують командний рядок (CLI), проєкти з графічним інтерфейсом (GUI) та складні застосунки, які використовують всі можливості ASP.NET.

На середовище CLI покладені задачі перевірки безпеки коду, виконання коду, виконання потоків, компіляція, керування пам’яттю та інші задачі. Для коду який виконується в середовищі CLI це є внутрішні середовища. В цілях безпеки, керованим компонентам присвоюються ступені довіри. Ступені довіри залежать від різних факторів, до яких входить і походження. Також середовище виключає появу багатьох проблем при розробці програмного забезпечення. Так наприклад виконується автоматичне розміщення об’єктів по посиланням та звільнення пам’яті коли об’єкт було видалено. Це дозволяє позбавитися найбільш частій проблемі додатків: витік пам’яті та недійсне посилання на пам’ять.

.NET Framework бібліотека класів представляє колекцію типі які взаємодіють з CLI та тісно інтегруються. Об’єктно орієнтована бібліотека надає типи від яких можна успадкувати виконуваний код. Це полегшує процес вивчення та набуття навичок при роботі та дозволяє легко об’єднувати компоненти сторонніх виробників з класами .NET Framework.

2.3 Порівняння Java та .NET Framework

Основна відмінність полягає у операційній системі. Java Enterprise Edition (JEE) може працювати на будь-якій системі в той час як .NET Framework працює в різних версія Windows. Все ж таки існують реалізації .NET з відкритим програмним кодом, які не націлені на Windows. Java завжди була кросплатформною та працювала на будь-якій системі, для якої існує відповідна віртуальна машина, роблячи її портативним та незалежним інструментом для розробки.

Обидві платформи підтримують низку різних мов програмування для забезпечення різноманітних функціональних можливостей. Це і PHP, Python, JavaScript, Ruby, Groovy, Scala, Clojure в той час як .NET Framework розрахований на мови сімейства Microsoft: C#, F#, VB.NET, C++. Платформа J2EE встановлює Java основною мовою, в той час як .NET дозволяє використовувати низку різних мов. Це додає гнучкості, хоча і вимагає від розробників більш різноманітних навичок кодування.

Досить важливим пунктом є інтегроване середовище розробника (IDE). Воно, як і будь який інструмент має бути зручним, високотехнологічним, та корисним в процесі написання коду. Сучасні середовища допомагають, підказують конструкції, пропонують низку різноманітних типів пошуку по коду, встановлення сторонніх плагінів, які покращують якісну складову процесу кодування та допомагають писати краще, зручніше та швидше. .NET Framework використовую в якості свого офіційного середовища продукт Microsoft Visual Studio. Розробники Java вільні обирати між чотирма основними IDE, запропоновані спільнотою.

- Eclipse від Eclipse Foundation.
- IDEA від Intellij.
- NetBeans від Oracle.
- JDeveloper від Oracle.

Середовища або безкоштовні, або мають безкоштовні дистрибутиви з обмеженими можливостями, або учбові безкоштовні ліцензії.

До переваг .NET можна віднести те, що мова C# проєктувалася маючи досвід Java та велика кількість проблем були вирішені. В цілому мови дуже схожі одна на одну. Друга вагома перевага є і недолік одночасно. .NET цілком і повністю підтримується Microsoft та всі програмні інструменти бібліотеки, фреймворки, середовища не рідко випускаються та оновлюються повільніше в середньому ніж у відкритих спільнот. Підтримка доступна тільки для Entity Framework що призводить до обмеженої об'єктно-реляційної підтримки .NET. Керований код працює повільніше ніж нативний та користувачі технології постійно залежать від розробників. З іншого боку, платформа забезпечує безперебійну горизонтальну масштабованість. Більшість інструментів та IDE можна знайти в мережі розробників.

До переваг Java можна віднести її портативність та кросплатформність. Супутні інструменти здебільшого безкоштовні та підтримуються велетенською спільнотою та постійно вдосконалюється. Java легко інтегрується з багатьма різними система завдяки багатому вибору готових інтеграційних рішень, бібліотек, фреймворків, наприклад Spring Integration. Міграція між платформами Java є легкою, оскільки вона забезпечує зворотню сумісність. Не є проблемою також інтеграція з базами даних, оскільки майже для всіх існують свої драйвери. Спільнота пропонує велику кількість різноманітних фреймворків та бібліотек для задоволення всіляких потреб. До недоліків можна віднести те що Java все ж таки кілька повільніше ніж інші мови програмування через використання віртуальної машини та байт-код, як проміжного коду між Java кодом та машинними інструкціями.

3. ОБГРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ

В ході виконання дипломного проєкту була обрана технологія Java та низка додаткових засобів для роботи веб-додатків. Система розрахована загалом на користувачів – відвідувачів кінотеатру. Вони повинні мати змогу зайти на ресурс, реєструватися, переглядати афішу, розклад сеансів, важливу інформацію, місця в залі, мати змогу забронювати квитки та повернути їх в разі необхідності. Також мусить бути спроектований функціонал адміністратора, який буде мати повноваження керувати ресурсами, афішою, розкладом, переглядати які користувачі коли і де саме забронювали місця в залі. Система має підтримувати кілька мов інтерфейсу. Для цього були обрані наступні технології:

- Мова Java.
- Spring Framework – багатофункціональний засіб для розробки веб-додатків.
- Мікросервісна клієнт-серверна архітектура.
- HTTP сервер Apache Tomcat.
- База даних MySQL.
- Пул з’єднань до бази даних Apache Commons DBCP.
- Фреймворк для модульного тестування Junit.
- Log4J для ведення логів.
- Bootstrap для інтерфейсу користувача.
- Інтегроване середовище розробки IDEA.

3.1 Клієнт-серверна архітектура.

Клієнт-серверна архітектура визначає спосіб основного обміну даними між комп'ютерами розташованими в різних місцях мережі. Клієнт-сервер це основний принцип у якого може бути кілька різних реалізацій. В основу покладено те що існує дві основні сторони: клієнт та сервер. Клієнт запитує дані у наперед визначеному форматі, а сервер в свою чергу приймає запит, готує дані та відправляє їх клієнту де вони розшифровуються, можуть бути додатково оброблені та відображені клієнту якщо це передбачено. Сервер глибоко стабільний та масштабований, що дозволяє йому оброблювати велику кількість клієнтів. Така архітектура зменшує трафік в Інтернеті, відповідаючи на запити клієнтів з підготовленими даними, замість пересилання цілих файлів сирих даних. Дані, в основному зберігаються в окремих базах даних, або файлових серверах. Сервер в свою чергу запитує дані в них. Клієнтом називається окрема програма яка здатна запитувати дані у сервера по мережі Інтернет, або локально, якщо це знаходиться в одній мережі. Це може бути як спеціалізований програмний продукт розрахований тільки для конкретного ресурсу, так і веб-браузер який запитує інформацію у вигляді гіпертексту по протоколу HTTP. В такому випадку сервери надають спеціалізований інтерфейс користувача для зручної взаємодії та запобіганню виникнення плутанини, проте якщо у ресурс підтримує стандартизований набір запитів, так званий API, то можна самому створити кілька різних інтерфейсів для взаємодії з одним і тим самим ресурсом. Це може бути зручно коли наприклад треба створити ще і додаток на мобільній платформі, який повинен працювати так само. Клієнти розташовані на окремих комп'ютерах, а сервер зазвичай в спеціалізованій потужній частині мережі. Також дані серверу можуть бути продубльовані в різних мережах в різних частинах світу для одночасного забезпечення потреб великої кількості користувачів та їх запитів.

Розрізняють два основні види взаємодії клієнт-сервер: дворівнева (рис 2) архітектура клієнт-серверної взаємодії та багаторівнева архітектура (рис. 3). Інколи багаторівневу називають трирівневою, але це просто частковий випадок багаторівневої.

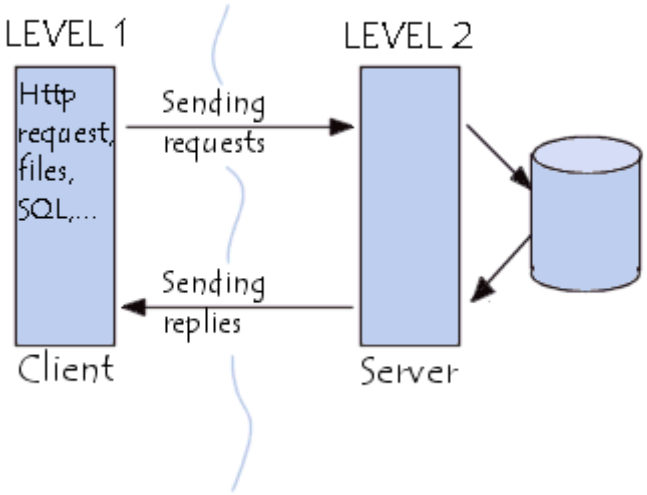

Рис. 2

При дворівневій архітектурі, кожен з рівнів представляють окремі комп'ютери розділені каналом зв'язку, зазвичай HTTP протоколом. Окремим програмним засобом є клієнт який відсилає запити та отримує відповіді від клієнта та сервер який комбінує в собі бізнес-логіку та сховище даних. У дворівневій архітектурі клієнт та сервер повинні обмінюватися даними безпосередньо. Якщо клієнт вводить дані, заповнює форму тощо, проміжних даних бути не повинно. Це робиться для забезпечення швидкості та уникнення плутанини між різними клієнтами посередниками. Так наприклад працюють системи онлайн-бронювання квитків.

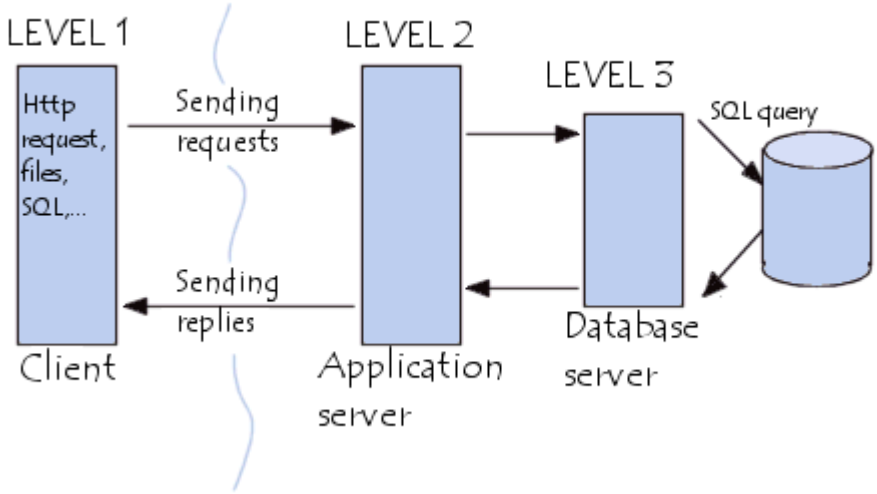

Рис. 3

Багаторівнева або трирівнева архітектура взаємодії відрізняється тим що другий рівень був розділений на два незалежних один з яких це сервер додатків, а другий сервер бази даних. В якості другого рівня виступає додаток який відповідає за виконання бізнес-логіки, обробку даних. Частіше за все це написана на PHP, Java, C#, Python програма, яка розташовується всередині якогось HTTP серверу та по цьому протоколу обмінюється даними з першим та другим рівнем. На третьому рівні розміщується система управління базою даних (СУБД). Другий та третій рівні можуть розміщуватись на одному комп'ютері, проте на практиці досить часто їх розносять на різні.

До переваг клієнт-серверної архітектури можна віднести те, що оскільки програма клієнта та сервера це незалежні програми, то комп'ютер клієнта володіє зниженими вимогами до обчислювальної здатності, оскільки всі найтяжчі обчислення будуть проводитися на серверній стороні. Також таке розділення покращує завадостійкість системи, а також гнучкість архітектури дозволяє адміністратору зробити локальну мережу більш захищеною.

До недоліків відноситься те, що серверне обладнання саме по собі коштує набагато більше за клієнтське та вимагає спеціально навченого

спеціаліста який буде адмініструвати та обслуговувати сервер. Якщо сервер «падає», або перенавантажується відкритими з'єднаннями, сесіями, трафіком то клієнти втрачають можливість працювати з даними.

Підсумовуючи описаний підхід треба сказати, що на принципі клієнт серверу сьогодні працює більшість програм в Інтернеті. Навіть «дектопні» програми мають функціонал для якого необхідний вихід в інтернет для обміну даними з віддаленим сервером, що робить дану технологію дуже широко розповсюдженою.

3.2 Мікросервіси та моноліт

Мікросервіси – архітектурна модель програмного забезпечення, яка може мати глибокий вплив не лише на функціональність IT підприємства, але і на цифрову трансформацію всього бізнесу. Мікросервіс у порівнянні з монолітною архітектурою являє собою фундаментальний прорив у підході IT до розробки програмного забезпечення та такого, який був успішно прийнятий такими організаціями, як Netflix, Google, Amazon та інші. Розглянемо переваги мікросервісної архітектури над монолітною.

Спочатку порівняємо мікросервісну та монолітну архітектуру. Монолітна програма будується як єдине ціле. Корпоративна програма складається з трьох частин: база даних (в якій міститься багато таблиць, як правило, в реляційній СУБД), інтерфейс користувача на стороні клієнта (що складається з HTML-сторінок та / або JavaScript, що працює в браузері), і на стороні сервера застосування. Цей серверний додаток буде обробляти HTTP-запити, виконуючи певну доменну логіку, отримувати та оновлювати дані з бази даних та заповнювати перегляди HTML, що надсилаються до браузера. Моноліт – єдиний виконуваний алгоритм. Щоб

змінити систему, розробник повинен створити та розгорнути оновлену версію програми на сервері.

В той час можливості мікросервісу визначаються за допомогою бізнес-орієнтованого API. Реалізація сервісу, яка може включати інтеграцію із системами зберігання даних, повністю приховано, оскільки інтерфейс визначений суто у відношенні бізнес речей. Позиціонування сервісів як цінних елементів для бізнесу неявно сприяє їх пристосуванню для використання в різних умовах. Один і той же сервіс може бути повторно використаний в кількох бізнес-процесах в залежності від потреби. Залежності між сервісами та тими хто їх використовує зведені до мінімуму, застосовуючи шаблон низького зчеплення (Low Coupling). Якщо контракти, виражені через API стандартизовані, бізнес зміни клієнтів не впливають на них не впливають.

Зазвичай процес розробки програмного забезпечення (waterflow, ітераційний тощо), як правило, передбачає велику команду розробників, які працюють над єдиним монолітним проєктом. Керівники проєктів, розробники та оперативний персонал можуть досягти різного ступеня успіху за допомогою цих моделей, особливо, коли вони набувають досвіду використання певного програмного забезпечення та інструменту розгортання. Однак існують деякі серйозні проблеми у використанні традиційних підходів:

- В процесі довгої розробки монолітного проєкту може виникнути ситуація, коли жоден розробник (або група розробників) не розуміє всієї програми.
- За визначенням, монолітні програми проектуються використовуючи єдиний інструмент розробки (JEE або .NET), який може обмежувати доступність.

- Важко досягти сприятливого темпу розробки при багаторазовому розгортанні монолітних проєктів.
- Масштабування монолітного проєкту часто буває великою проблемою.
- Обмежене повторне використання.

Мікросервісна архітектура в поєднанні з хмарними технологіями, управлінням API та технологіями інтеграції створює інший підхід до розробки програмного забезпечення. Натомість моноліт розподіляють на низку незалежних служб, які розробляються та підтримуються окремо. Це має такі переваги:

- Розробка сервісів розподіляє логіку та дозволяє розробляти знаходячись в рамках окремого технічного завдання.
- Сервіси повинні мати невеликі розміри. Зазвичай розраховані на підтримку невеликою кількістю розробників.
- Сервіси існують як артефакти які можна незалежно розгортати на сервері і їх можна масштабувати незалежно від інших служб.
- Мікросервісів обмінюються даними за допомогою стандартних протоколів, наприклад, API REST, він може використовуватись іншими службами та додатками без прив'язки до мови або спільних бібліотек.

Мікросервіси мають суттєву перевагу над монолітом з точки зору бізнесу. При правильному розгортанні мікросервіси можуть принести значну користь бізнесу. Ця користь може бути як підвищення ефективності так і зниження технічного боргу.

Зниження технічної заборгованості – одна з переваг, але не єдина, яку пропонують мікросервіси; проте помірна економія не закінчується лише технічною заборгованістю. Мікросервіси мають інші переваги, які можуть знизити витрати та вплинути на результат, а саме:

- Використання мікросервісів може призвести до підвищення ефективності використання коду та інфраструктури. Досить часто вдається знизити витрати на 50%, зменшивши кількість інфраструктури, необхідної для запуску даної програми.
- Розділяючи функціональність на рівні, а потім абстрагуючи відповідні сервіси, DevOps спеціаліст може зосередитись лише на оновленні відповідних фрагментів програми. Це допомагає уникнути тяжкого процесу інтеграції, притаманному монолітним проєктам. Розвиток мікросервісів швидко перетворює його на процес, який можна здійснити за тижні, а не місяці.
- Гнучкість. Розповсюджуючи функціональних можливостей у кількох сервісах, можна зменшити сприятливість до однієї точки відмовиЯк результат ми отримуємо додатки, які працюють краще, мають менший час простою та за потреби масштабуються.
- Менший час простою та можливість ітераційного процесу розробки сприяє збільшенню доходу. Утримання уваги користувачів збільшується завдяки впровадженню постійних удосконалень, які можливі завдяки використанню мікросервісів.

3.3 REST та RESTful

REST (Representational state transfer) – Різновид стилю архітектури програмного забезпечення який використовується для побудови веб-служб в розподілених системах, таких як World Wide Web. Рой Філдінг, один із авторів протоколу HTTP увів термін REST в 2000 році. Системи які повністю відповідають всім характеристиками REST називаються RESTful системами. Кожній одиниці інформації відповідає свій URL ресурс. Використовуючи в повній мірі протокол HTTP та його методи, можна спроектувати систему в якій всі дані, отримання, вставка, видалення, будуть відповідати конкретні HTTP ресурси. Таким чином URL по суті є первинним ключем для одиниці даних. Приводячи приклад, отримання умовно третьої книги буде мати ресурс /book/3, а отримання 12 сторінки відповідно /book/12. Специфікація REST не вимагає HTTP або JSON. Специфікація взагалі не згадує JSON або XML. Найбільша перевага таких сервісів, що з ними може працювати будь-яка система, здатна «парсити» JSON повідомлення або XML файли та виконувати запити по HTTP.

Архітектура REST має багато переваг. Рой Філдінг розробив його для Інтернету, більшість обмежень які він прогнозував і через 18 років все ще є. У 2000 році не існувало Android чи iPhone. Internet Explorer мав 50% ринку браузерів. Найбільшим конкурентом став Mozilla Firefox. Але Філдінг визнав, для чого потрібні онлайн-додатки та як веб-клієнти поступово перетворилися з окремих платформ відображення HTML на повноцінні програми. Інструменти, якими ми користуємось сьогодні, виросли, щоб відповідати REST, а не навпаки.

3.4 Spring Framework

До того як з’явилася технологія Enterprise Java Beans (EJB) розробникам необхідно було використовувати JavaBeans для розробки веб-додатків. У розробці компонентів інтерфейсу користувача (UI) широко використовувався JavaBeans. Розробники не мали надійних відомих інструментів для розробки надійних та безпечних корпоративних програм. З появою EJB стало можливе розширення веб та корпоративних компонентів Java та надавати послуги, у розробці Enterprise проєктів. Розробка програми за допомогою EJB була досить складною та ресурсоємкою, оскільки потрібно було виконувати різні завдання, такі як створення інтерфейсів для домашнього та віддаленого користування та реалізація методів зворотного виклику життєвого циклу, що призводить до росту складності.

Для вирішення великої кількості набутих незручностей та проблем було розроблено Spring Framework. В ньому використовуються різні нові методи та підходи, такі як, аспект-орієнтоване програмування (AOP), POJO (Plain Old Java Object) та DI (Dependency Injection), для розробки корпоративних програм, тим самим уникнення складностей, пов'язані з розробкою програм за використовуючи EJB. Новий фреймворк з відкритим кодом, який дозволяє розробникам JavaEE надійні корпоративні програми. Здебільшого фреймворк допомагає вам керувати своїми бізнес-об'єктами. Це значно спростило розробку веб-додатків порівняно з класичними фреймворками Java та інтерфейсами API, такими як підключення до бази даних, JSP сторінки та сервлетами.

Spring це не один суцільний, самодостатній фреймворк. Він в себе включає велику кількість відокремлених спеціалізованих проєктів поменше таких як Spring Data, Spring AOP, Spring Integration, Spring ORM, Spring

MVC. Кожен з цих модулів можна створювати окремо під час створення. Щоб забезпечити кращі функціональні можливості модулі легко групуються разом.

Наявність таких особливостей як IoC (Inversion of Control) контейнер, аспекти, управління транзакціями, робить цей фреймворк унікальним. Ось короткий перелік головних особливостей:

- IoC контейнер. Це э ядро фреймворку. В контейнері створюються так звані біни, джава об’єкти, серсіви, контролери тощо, і контейнера самостійно визначає залежності між ними та виконує зв’язування (Dependency Injection).
- Spring Data. Більш високорівневий шар взаємодії з базою даних. По замовчуванню використовує Hibernate та забезпечує дуже зручне та високорівневе зберігання об’єктів в базі даних.
- Spring MVC. Реалізація шаблону Model-View-Controller. Дозволяє створювати додатки використовуючи розділення обов’язків між обробником запитів (контролер), бізнес-логікою (моделлю) та відображенням (view). Всі запити спочатку надсилаються в контролер, а потім оброблюються окремою JSP сторінкою або сервлетом. Легко інтегрується з іншими технологіями користувацького інтерфейсу.
- Управління транзакціями. Виконує генерацію кінцевих точок та визначає веб-служби Java. Spring пропонує багатошарові підходи, якими керується розбір мови розмітки XML. Spring забезпечує ефективне відображення для передачі вхідного запиту XML-повідомлення та можливість легкого розподілу XML-повідомлення між двома клієнтами.

Spring Framework архітектурно складається з семи модулів (Рис. 4).

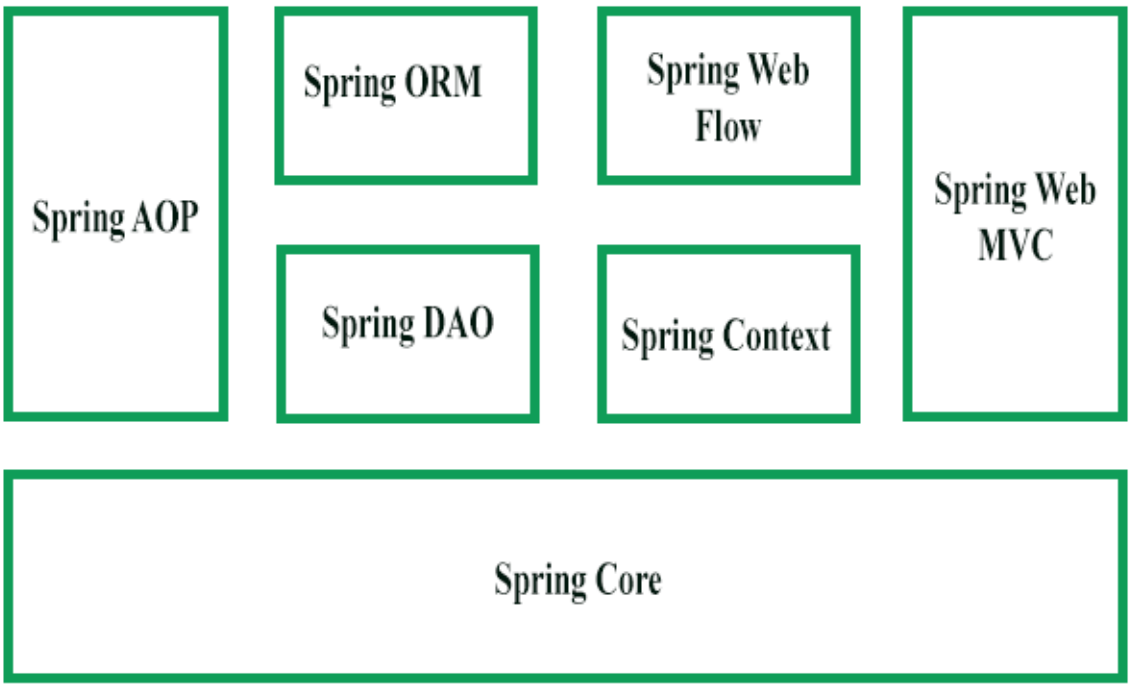

Рис 4.

Це модулі Spring Core, Spring Web MVC, Spring AOP, Spring DAO, Spring ORM, Spring Context та Spring Web Flow. За допомого. Цих модулів можна розробляти найрізноманітніші веб-додатки; наприклад, можна користуватися модулем Spring MVC для розробки програм на основі MVC шаблону.

Підсумовуючи, треба сказати, що Spring Framework сьогодні це дуже потужний інструмент в руках розробників веб-орієнтованих додатків який пропонує широкий вибір різних модулів для задоволення більшості вснуючих потреб бізнесу. Сьогодні Spring це один із найпопулярніших фреймворків для Java розробників.

3.5 Система управління базами даних MySQL.

База даних - це структурований сховище даних різних типів. Це може бути все, від простого списку покупок до галереї зображень або величезної кількості інформації у корпоративної мережі. Для додавання, доступу та обробки даних, що зберігаються в комп'ютерній базі даних, вам потрібна система управління базами даних, наприклад MySQL Server. Оскільки комп'ютери дуже добре обробляють велику кількість даних, системи управління базами даних відіграють центральну роль у обчисленні, як автономні утиліти, так і як частини інших додатків.

База даних э реляційна. Це значить що дані зберігаються в окремих таблицях, а не в одному великому сховищ, як наприклад в NoSQL базах даних. Дані зберігаються в фізичних файлах, що в комбінаціїї з алгоритмами оптимізації дає великий приріст в швидкості взаємодії. Дана модель, в якій є бази даних, таблиці, рядки, стовпці пропонує гнучке середовище програмування. Реляційна модель дозволяє встановлювати правила, по яким дані одне з одним співвідносяться. Такі відношення, як один до одного, один до багатьох, унікальне поле, обов’язкове та необов’язкове тощо. Таким чином при виконаний умовах нормалізації, програма ніколи не бачить непослідовних даних, дублікатів, застарілих або відсутніх даних.

SQL - стандартизована мова запитів. Декларативна мова за допомогою якої можна запитувати ті дані які потрібно та у вказаній формі. Можна вводити SQL безпосередньо наприклад, для генерації звітів. Можна вставляти SQL заяви в код, написаний іншою мовою, або використовувати специфічний для мови API, який приховує SQL.

MySQL це окремий сервер який працює в режимі клієнт-сервер. Він виконується окремим процесом в операційній системі, та може бути

розміщеній на іншій машині в мережі, оскільки обмін даними відбувається по протоколу TCP/IP.

MySQL використовує за замовчуванням рушій InnoDB який керує системою управління базами даних та забезпечує баланс між надійністю даних та швидкістю роботи. InnoDB є рушієм зберігання даних MySQL за замовчуванням. До основних переваг можна віднести:

- Підтримка DML (мова маніпулювання даними) - операції відповідають моделі ACID, транзакції, що виконують фіксацію, відкат та відновлення даних після аварійних ситуацій.
- Забезпечення блокування на рівні рядків та послідовне зчитування у стилі Oracle збільшують ефективність та роботу при великій кількості користувачів.
- Забезпечення впорядкування даних таблиць на диску для оптимізації запитів на основі первинних ключів . Кожна таблиця неявно індексується по первинному ключу, який називається кластерним індексом, за допомогою чого дані організовуються для мінімізації вводу / виводу для пошуку первинних ключів.
- Підтримка зовнішніх ключів FOREIGN KEY для забезпечення цілісності даних. Зовнішні ключі, вставки, оновлення та видалення перевіряються, щоб не призвести до невідповідностей у різних таблицях.

InnoDB підтримує модель ACID, набір головних принципів проєктування баз даних, які описують положення надійності, які важливі для програм та безпеки даних. Виняткові умови не спотворюють дані та не пошкоджують результати. InnoDB дотримується моделі ACID та попереджує проблеми пов’язані зі збоєм програмного забезпечення та несправністю обладнання. Якщо покладатися на функціональні можливості, сумісні з

ACID, тоді не потрібно розробляти особистий функціонал перевірки узгодженості та механізми відновлення після аварійних ситуацій.

Основні принципи ACID складаються з:

- A – Atomicity – Атомарність.
- C - Consistency – Узгодженість.
- I - Isolation – Ізоляція.
- D – Durability – Довговічність.

Передбачено наступні операції для забезпечення атомарності:

- Виконання операції закріплення даних commit.
- Виконання операції відкату транзакції rollback.
- Налаштування атоматичного закріплення даних.

Передбачено наступні операції для забезпечення узгодженості:

- Подвійний запис буферу.
- Відновлення після аварії.

Передбачено наступні операції для забезпечення ізоляції:

- Налаштування атоматичного закріплення даних.
- Встановлення рівню ізоляції транзакцій.
- Налаштування деталей низького рівня InnoDB

Передбачено наступні операції для забезпечення довговічності:

- Подвійний буфер, може бути увімкнутий опцією innodb_doublewrite.
- Конфігурація innodb_flush_log_at_trx_commit.
- Конфігурація sync_binlog.
- Конфігурація innodb_file_per_table.
- Записування буферу на дисковий накопичував.

- Підтримка кешу.
- Створення резервних копій та реплікація даних.

Реплікація MySQL працює для таблиць. Реплікація даних може відбуватися навіть якщо рушій на підлеглій СУБД відрізняється від того який використовується в головній системі. Таким чином можна скопіювати таблиці InnoDB головного кластеру до MyISAM на підлеглому. Транзакції, які провалюються на головній одиниці, взагалі не впливають на реплікацію. MySQL використовує двійковий лог для реалізації реплікації, куди записує оператори SQL, які змінюють дані. Транзакція, яка не спрацювала не записується у бінарний лог, і як результат не виконується на підлеглих кластерах.

Каскадні дії для таблиць на головному кластері копіюються на підлеглий, лише коли таблиці які пов’язані зовнішнім ключем на обох кластерах, на головному та підлеглому використовують InnoDB. Це справедливо, якщо використовується реплікація на основі «стейтментів» чи рядків.

4. ОПИС РОЗРОБКИ ДОДАТКУ

4.1 JWT аутентифікація

JWT – Json Web Token – стандарт який описує формат та основні положення створення токенів доступу на основі JSON. Широко використовується в цілях передачі аутентифікаційних даних у клієнт-серверних додатках. Сервер створює та підписує токен, після чого передає його клієнту, який використовує його для підтвердження своєї особистості під час доступу до захищених ресурсів, які потребують авторизації.

Найпростішим підходом авторизації у REST архітектурі могло бути надсилання логіну та паролю кожен раз коли клієнт намагається отримати данні із захищеного ресурсу. Проте такий підхід не є безпечним, особливо коли використовується незахищений протокол. Кращим рішенням є надання деякого токену, який створюється на основі логіну та паролю деякою хеш функцією та видається користувачу (Рис. 5). В базі даних зберігається пара токен – id. Клієнт просто надає цей токен кожен раз, а сервер перевіряє чи існує такий токен, якщо так то користувача авторизують. Також токену з метою безпеки надається термін дії після якого він стає недійсний.

В проєкті JWT аутентифікація та авторизація реалізована за допомогою засобів суміжного проєкту Spring Security з можливістю встановлення ролей та прав. Таким чином в токені міститься зашифрована інформація про логін, пароль та права доступу користувача.

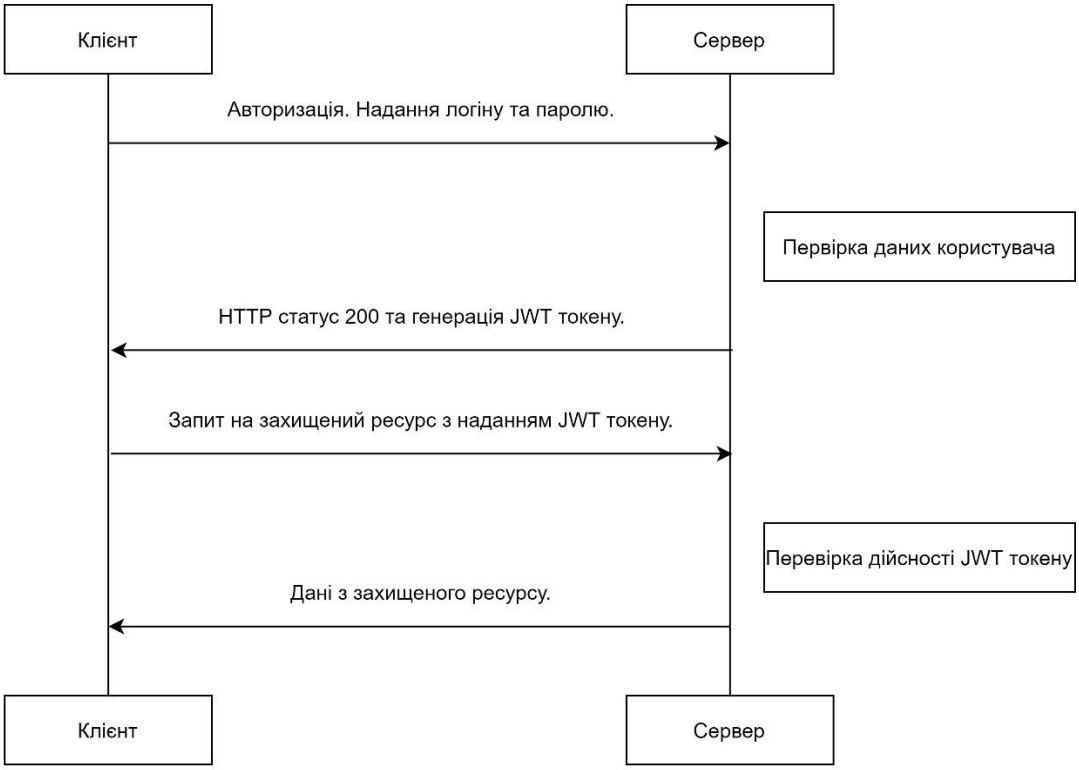

Рис.5. Процедура створення токену.

При спробі доступитися до захищеного ресурсу без надання токену – отримуємо помилку (Рис. 6).

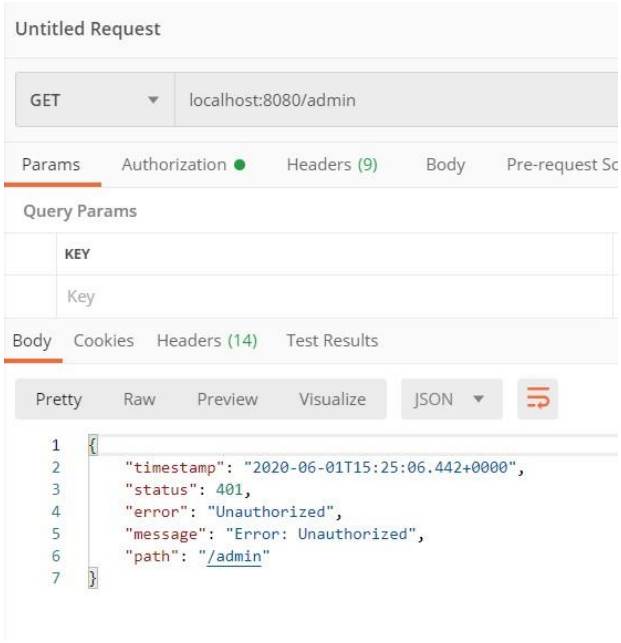

Рис. 6

Вказавши в запиті логін та пароль у форматі JSON по ресурсу /signin отримуємо згенерований токен у разі дійсності даних (Рис. 7).

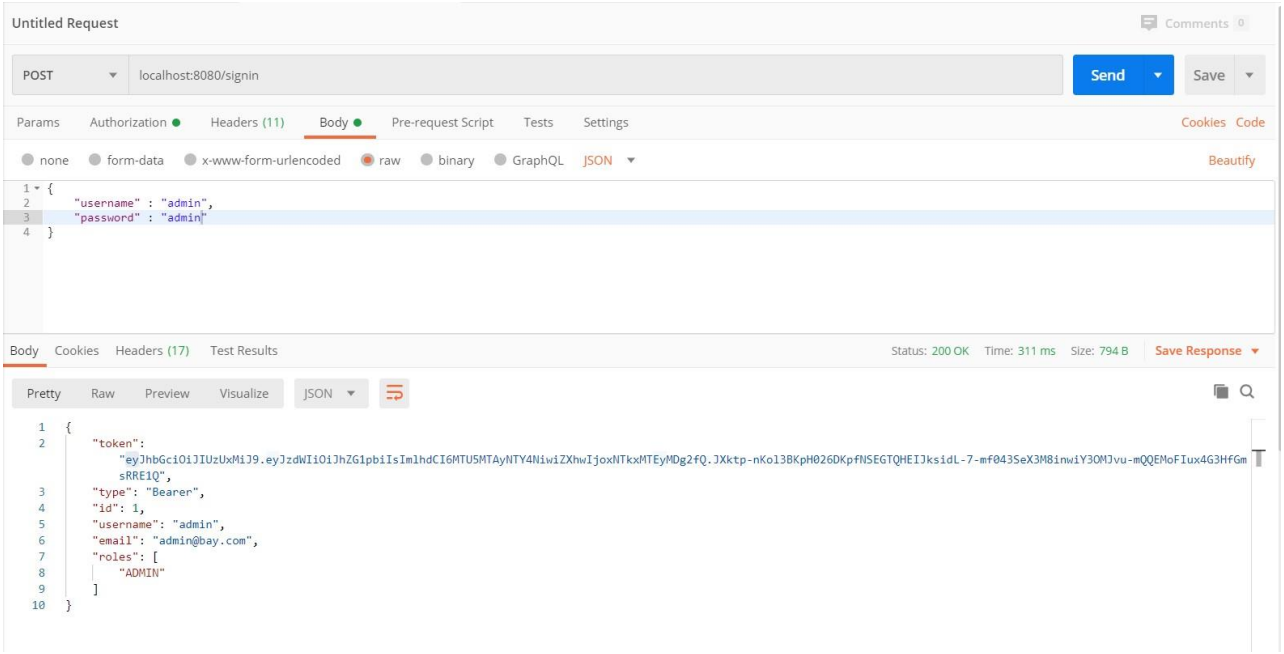

Рис. 7

Вказавши токен у авторизаційному полі запиту отримуємо доступ до захищених ресурсів (Рис. 8). Таким чином можна створювати SPA – односторінковий додаток клієнтської сторони.

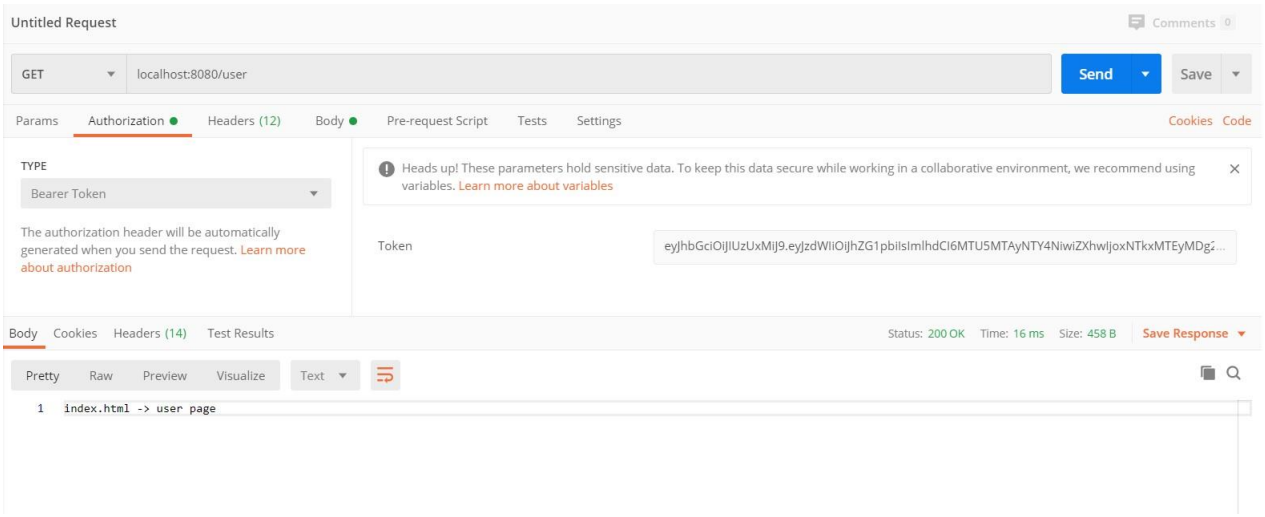

Рис. 8

4.2 Організація даних

Дані зберігаються в СУБД MySQL та зберігаються в восьми різних таблицях, пов’язаних зовнішніми ключами. Кожна таблиця, окрім days_translate та movies_translate зберігає дані, закріплені за конкретною об’єктною сутністю. Таким чином в таблиці sessions зберігаються сутності сесій які пов’язані з фільмами (один до одного), днями (один до одного), та білетами (один до багатьох). Наступні таблиці зберігають такі дані:

- users – дані про користувачів.
- movies – дані про фільми.
- tickets – дані про білети.
- sessions – дані про сеанси.
- days – дані про дні.
- languages – мови.
- movies_translate – інформація про фільми яка підлягає перекладу.
- days_translate – назви днів які підлягають перекладу.

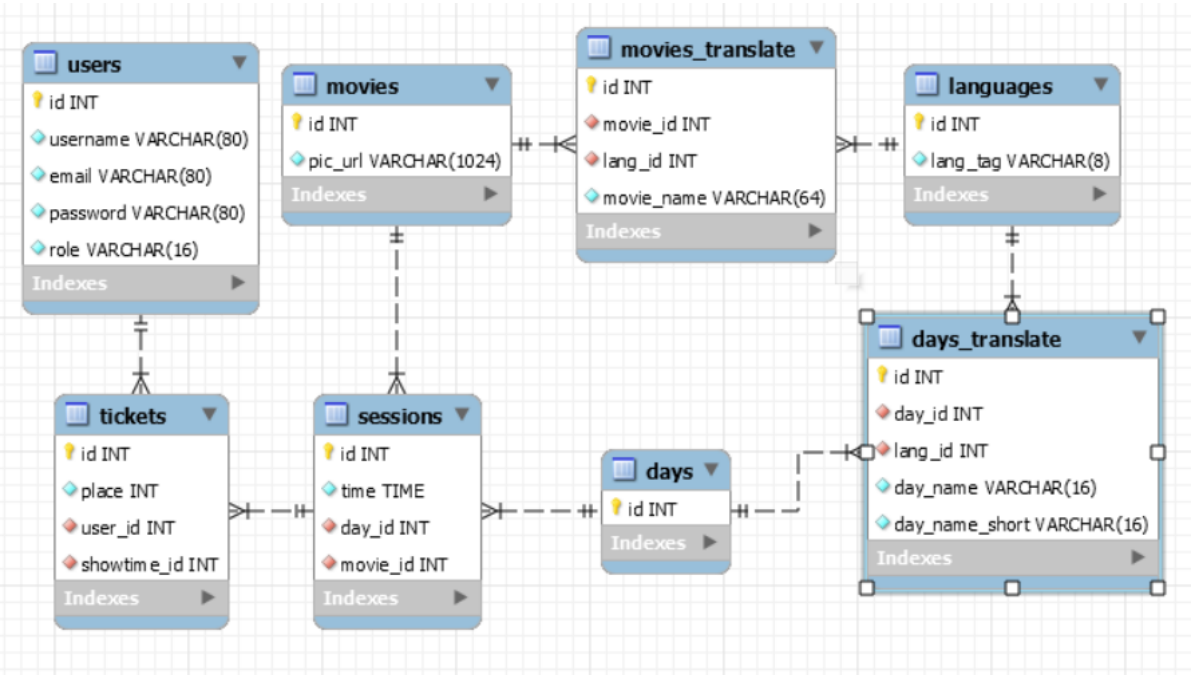

Рис. 9

4.3 Розробка інтерфейсу

Інтерфейс користувача повинен бути достатньо простим, не переповненим багатьма непотрібними елементами, тобто бути дещо мінімалістичним, інформативним та відповідати сучасним тенденціям веб-дизайну. Недопустиме використання застарілих елементів дизайну минулих десятиліть. Також по можливості сторінки не повинні бути навантажені ресурсовитратними медіа елементами, відеозаписами тощо.

Сторінки створені використовуючи верстку типу flexbox що дозволяє зробити гнучкі адаптивні до різних форматів вікон конструкції. На головній сторінці розташовується header з посиланнями на головну сторінку, афішу, сеанси, вибір мови, реєстрацію та вхід. Посередині – великий слайдер з постерами фільмів, назвою, головними акторами та режисером. Трохи нижче розташований footer. Вигляд головної сторінки для неатворизованого користувача (Рис 10).

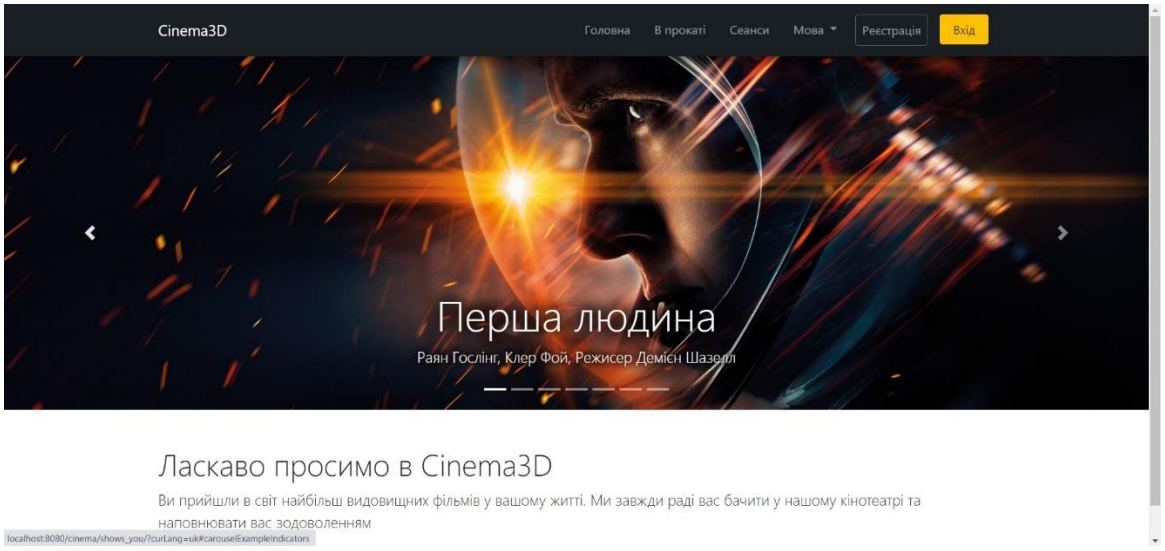

Рис. 10

Натиснувши на кнопку «Вхід» відкривається панель авторизації з полями для введення логіну та паролю. Замість логіну може бути введена електронна пошта. Відправка запиту на сервер відбувається асинхронно,

використовуючи технологію AJAX, яка дозволяє відправляти запит без перезавантаження сторінки. Під кнопкою «Вхід» в панелі авторизаціїї розташоване поле відображення помилки в разі неправильного вводу даних.

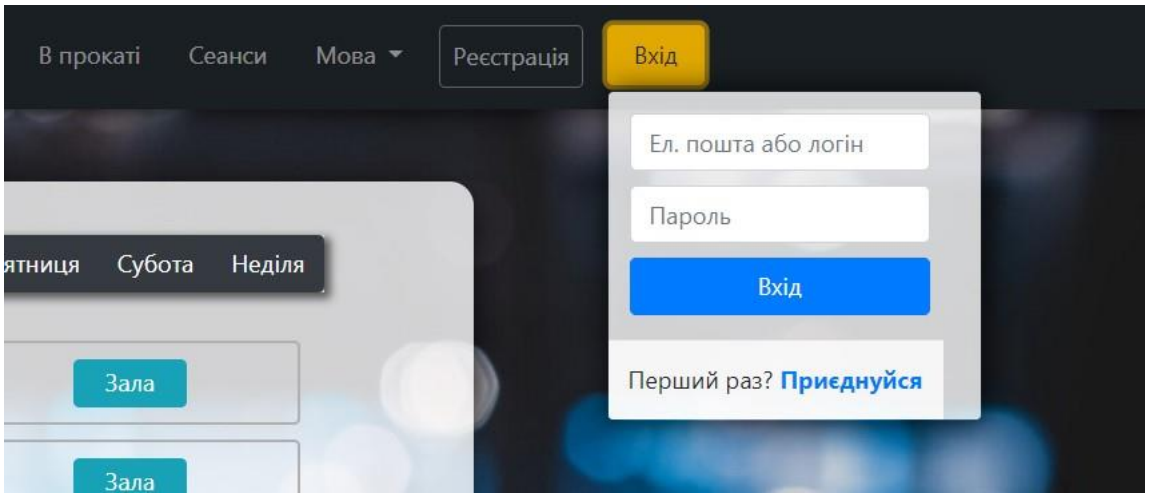

Рис. 11

У випадаючому вікні можна вибрати мову інтерфейсу. Проєкт підтримує декілька мов з можливістю розширення (Рис. 12).

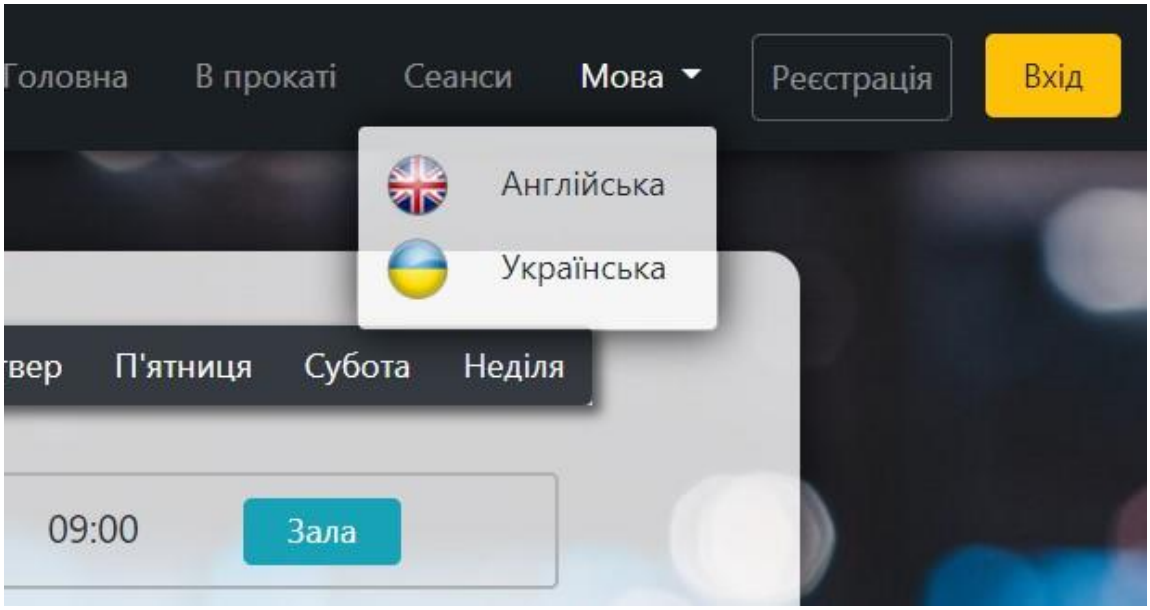

Рис. 12

Афіша виконана у вигляді каруселі, де кожним елементом є картка з картинкою, назвою та переліком наявних сеансів день, година. Кожен сеанс це кнопка, тому можна натиснути і потрапити у кімнату цього сеансу (Рис. 13).

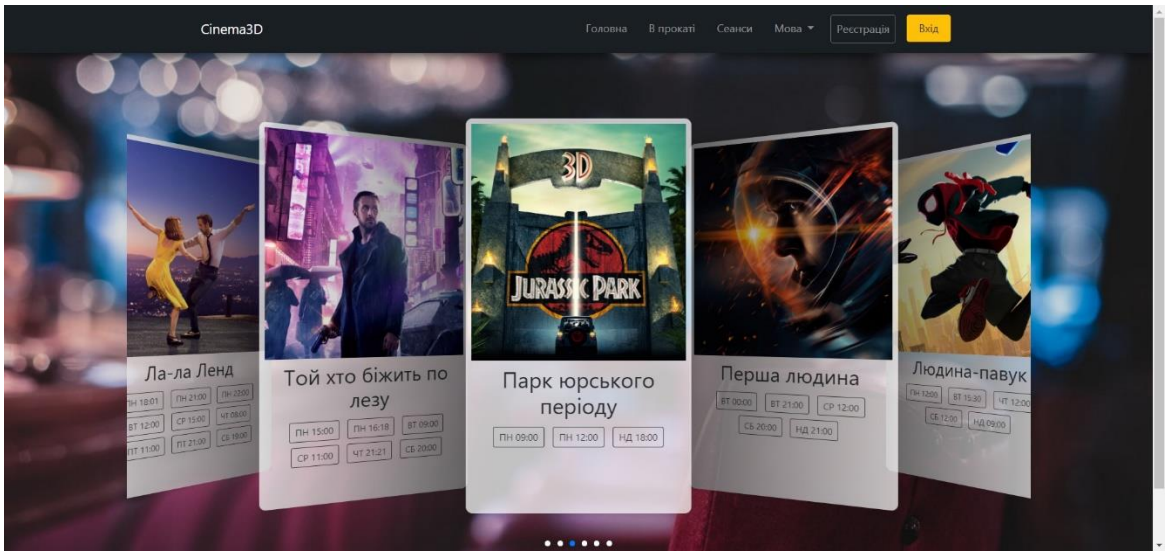

Рис. 13

Також сеанс можна знайти на відповідній сторінці, на якій вони відображені у вигляді списку де можна вибрати день та побачити перелік наявних сеансів фільмів, час та заповненість зали, натиснувши на кнопку (Рис. 14).

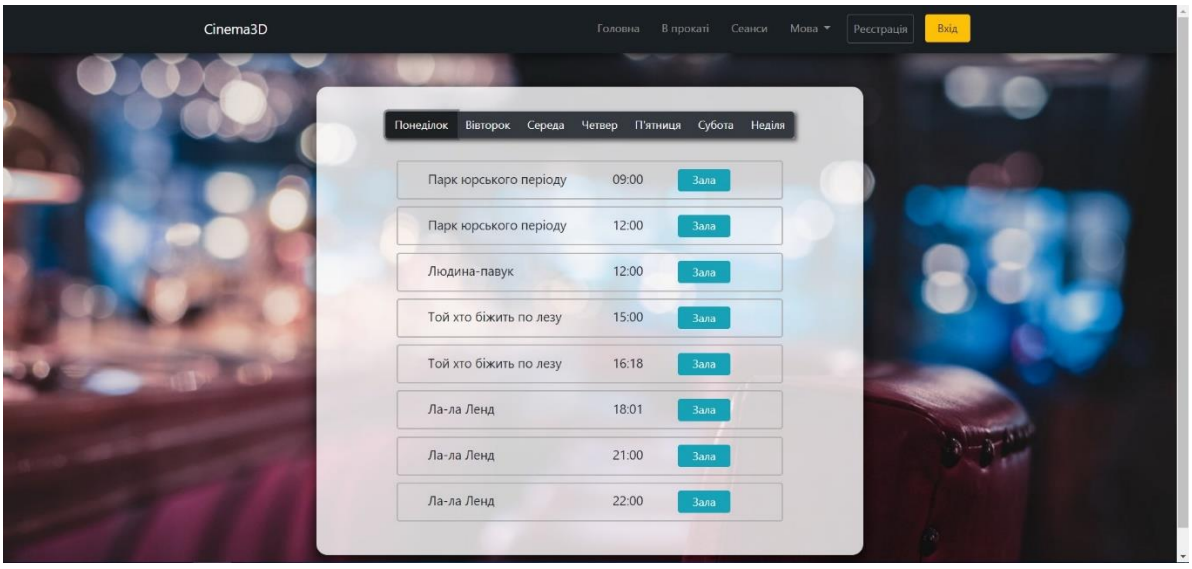

Рис. 14

Для кожного сеансу можна переглянути заповненість зали. Зареєстровані користувачі можуть купити квиток. Для незареєстрованих доступний тільки перегляд (Рис. 15).

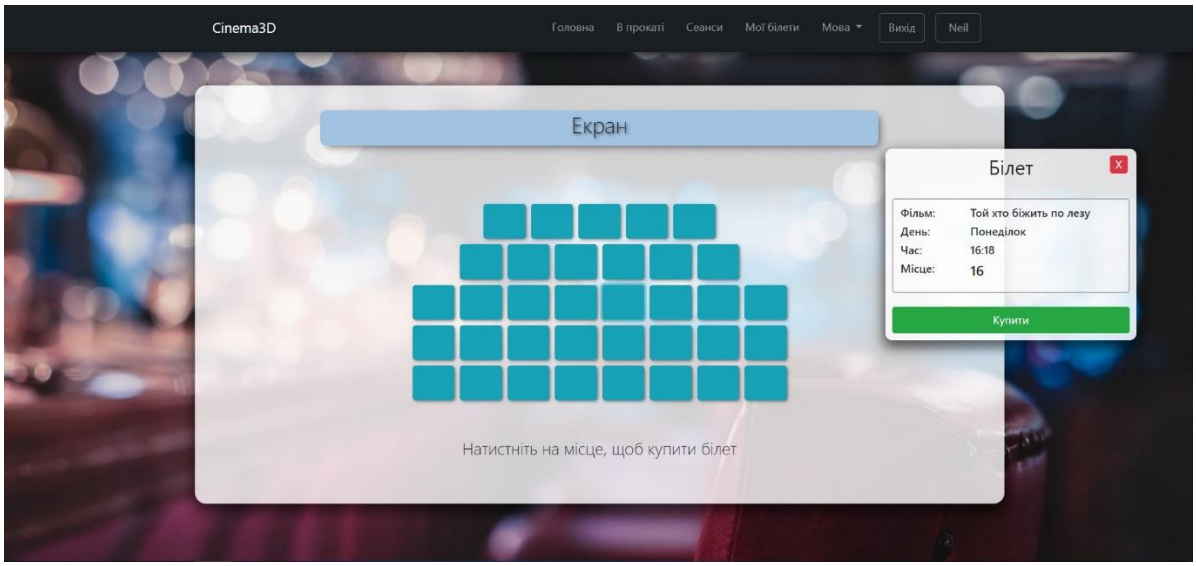

Рис. 15

На сторінці реєстрації розміщені звичні поля які необхідно заповнити. Логін, пароль, електронна пошта, кожне з яких проходить перевірку регулярним виразом на правильність вводу. Також реалізованно з використанням технології Ajax. Можна вибрати мову та повернутися на головну сторінку (Рис. 16). Зареєстровані користувачі можуть переглядати список з квитками, закріпленими за їх акаунтом. Відповідне посилання наявне в головному «хедері» сайту. Можна повернути квиток натиснувши кнопку. Кожен квиток пронумерований та має свій id. За ним закріплений фільм, сеанс, час та місце в залі (Рис 17).

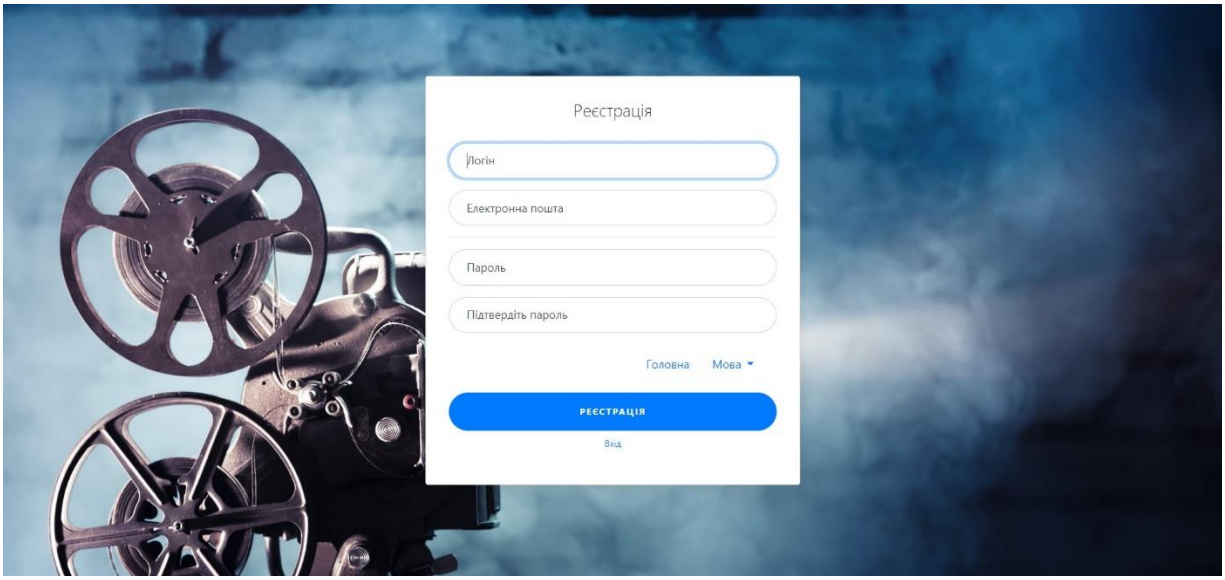

Рис. 16

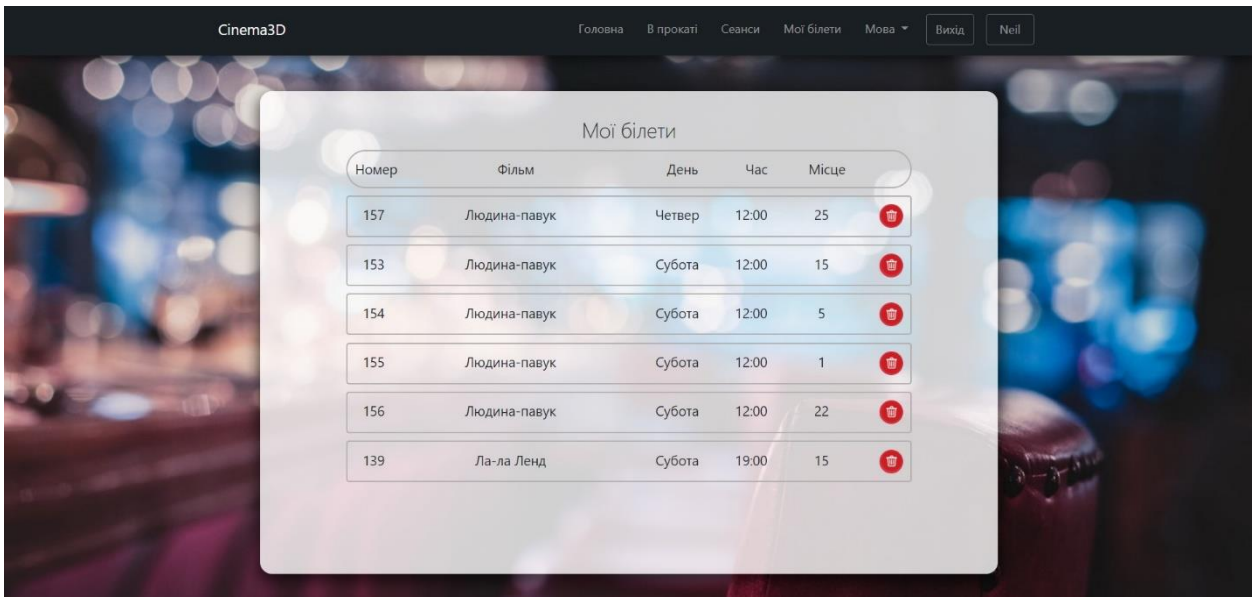

Рис. 17

Додаток має підтримку адмін-клієнту. Свого роду суперкористувач який має змогу редагувати афішу та список сеансів, створюючи та видаляючи відповідні елементи. Можна додати фільм, прикріпивши картинку, назву двома мовами, або видалити (Рис. 20). Таким самим чином процес відбувається з сеансами (Рис. 19). Адміністратор отримує попередження, що квитки клієнтів зникнуть, якщо хоча б один вже заброньований.

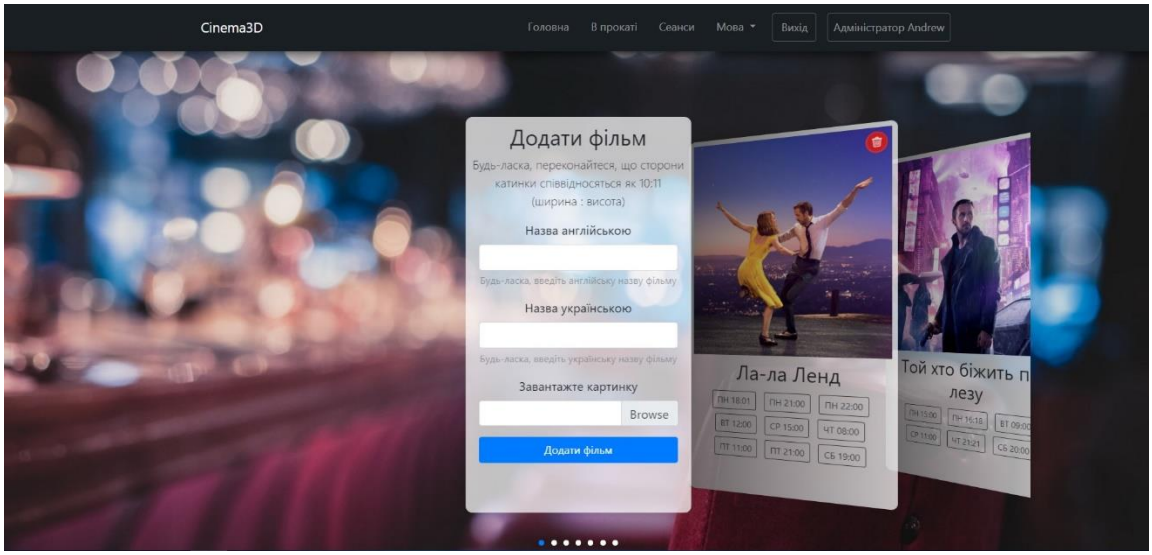
Cinema3D
Головна
В прокаті
Сеанси
Мова
Вихід
Адміністратор Andrew
Додати фільм
Будь-ласка, переконайтеся, що сторони
катинки співвідносяться як 10:11
(ширина : висота)
Назва англійською
Будь-ласка, введіть англійську назву фільму
Назва українською
Будь-ласка, введіть українську назву фільму
Завантажте картинку
Browse
Додати фільм
Ла-ла Ленд
Той хто біжить по лезу

Рис. 18

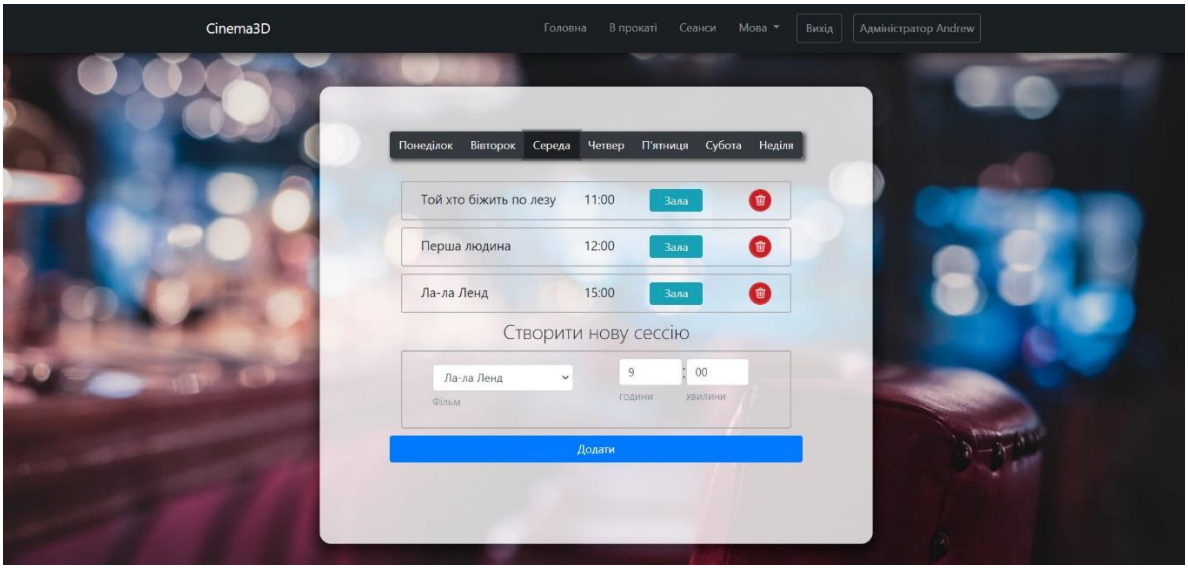
Cinema3D
Головна
В прокаті
Сеанси
Мова
Вихід
Адміністратор Andrew
Понеділок
Вівторок
Середа
Четвер
П'ятниця
Субота
Неділя
Той хто біжить по лезу
11:00
Зала
Перша людина
12:00
Зала
Ла-ла Ленд
15:00
Зала
Створити нову сессію
Ла-ла Ленд
Фільм
9
00
години
хвилини
Додати

Рис. 19

5. НОВИЗНА ВИКОРИСТАНИХ РІШЕНЬ ТА ПІДХОДІВ

5.1 Реактивність. Spring WebFlux

Фреймворк Spring 5 використовує бібліотеку Java Project Reactor яка реалізує модель реактивного програмування на основі специфікації реактивних потоків для створення реактивних додатків. В Spring Framework 5 включено модуль spring-webflux. В модулі міститься код який підтримує реактивні клієнти HTTP, веб-сокетів, реактивних веб-додатків, а також REST.

Реактивне програмування це неблокуючі асинхронні програми, які керуються подіями та потребують невеликої кількості потоків для вертикального масштабування на відміну від горизонтального.

Основна ідея реактивності це асинхронність. Асинхронність це подійно орієнтована концепція в якій діями керують наявність даних в конкретний момент часу. В асинхронності має місце практично повна аналогія з повідомленнями в месенджері чи callback і promises в JavaScript. Можна створювати потоки даних з подій. Реактивний підхід зменшує рівень абстракції додатку в цілому дозволяючи концентруватися лише на бізнес логіці, замість того, щоб підтримувати проєкт з великою кількістю допоміжних деталей. Код з використанням асинхронних принципів буде помітно менший, а значить і простіший для розуміння та підтримки. А також збільшується швидкість роботи систем, завдяки використанню неблокуючого API. Реактивне програмування добре підходить для обробки великої кількості різноманітних подій.

ВИСНОВКИ

В ході виконання дипломного проєкту було розроблено корпоративний веб-додаток для системи кінотеатру. В процесі проєктування були використані сучасні засоби такі, як стек технологій Java та фреймворк Spring Framework. Була використана прогресивна архітектурна модель мікросервісів, які дозволяють розробляти програмне забезпечення поєднуючи незалежні веб-сервіси. Також це надає можливість використовувати різні технології, різні мови, виконуючи обмін даними за допомогою стандартного стеку протоколів та HTTP викликів API, а розподіл на незалежно розгорнуті компоненти забезпечує гарну масштабованість, стабільність та керованість.

В розробленій системі використані можливості модулю реактивного програмування WebFlux фреймворку Spring. Застосовуючи асинхронний ввід/вивід досягається значне якісне збільшення навантажувальної здатності системи. Реактивність дозволяє ефективно оброблювати операції які передбачають затримки, такі як запит даних з віддаленого сервера; одночасно обслуговувати значно більшу кількість запитів; забезпечити кращу якість обслуговування та передбачувано планувати пропускну спроможність.

Розроблена система забезпечує великий потенціал для модернізації та розширення функціоналу, що є головною властивістю бізнес проєкту. В подальшому може бути розширений перелік ролей користувачів; впроваджені інші відомі способи авторизації, такі як OAuth2.0; виконана інтеграція зі сторонніми системами; зберігання даних у хмарних сервісах; додавання підтримки відео трейлерів на сторінках фільмів тощо.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Web Reactive Stack – Електрон. дані (1 файл) – Режим доступу: https://docs.spring.io/spring/docs/current/spring-framework-reference/web-reactive.html (дата звернення 23.05.2020) – Назва з екрану.
2. Україна кінотеатр - Електрон. дані (1 файл) – Режим доступу: http://kinoukraina.zt.ua/ (дата звернення 21.05.2020) – Назва з екрану.
3. Кінотеатр «Баттерфляй» - Електрон. дані (1 файл) – Режим доступу: https://www.kino-butterfly.com.ua/ (дата звернення 22.05.2020) – Назва з екрану.
4. КІНО-ТЕАТР.UA – Електрон. дані (1 файл) – Режим доступу: https://kino-teatr.ua/ (дата звернення 12.05.2020) – Назва з екрану.
5. .NET vs Java: Unbiased Comparison - Електрон. дані (1 файл) – Режим доступу: https://scand.com/company/blog/net-vs-java-comparison/ (дата звернення 29.05.2020) – Назва з екрану.
6. MySQL Documentation – Електрон. дані. – Режим доступу: https://dev.mysql.com/doc/ (дата звернення 24.05.2020). – Назва з екрану.
7. Краткий обзор движков таблиц MySQL – Електрон. дані (1 файл). – Режим доступу: https://habr.com/ru/post/64851/ (дата звернення 29.05.2020). – Назва з екрану.
8. Як використовувати JSON Web Token (JWT) для аутентифікації – Електрон. дані (1 файл). – Режим доступу: https://codeguida.com/post/1567 (дата звернення 29.05.2020). – Назва з екрану.
9. О модели взаимодействия клиент-сервер простыми словами. Архитектура «клиент-сервер» с примерами – Електрон. дані (1 файл). – Режим доступу: https://zametkinapolyah.ru/servera-i-protokoly/ (дата звернення 29.05.2020). – Назва з екрану.